UNCLASSIFIED
URITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No 0704-0188

REPORT SECURITY CLASSIFICATION NCLASSIFIED	1b RESTRICTIVE MARKINGS
SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION / AVAILABILITY OF REPORT Approved for public release; distribution is unlimited
DECLASSIFICATION / DOWNGRADING SCHEDULE	
PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
NAME OF PERFORMING ORGANIZATION aval Postgraduate School	6b OFFICE SYMBOL (If applicable) EC
7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School	
ADDRESS (City, State, and ZIP Code) onterey, CA 93943-5000	7b ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000
a. NAME OF FUNDING / SPONSORING ORGANIZATION	8b OFFICE SYMBOL (If applicable)
9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
c. ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO	PROJECT NO	TASK NO	WORK UNIT ACCESSION NO

1. TITLE (Include Security Classification)
PPLICATION OF KALMAN FILTER ON MULTISENSOR FUSION TRACKING

2 PERSONAL AUTHOR(S)
'ERPENING, Brian Everett

3a. TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
laster's Thesis	FROM _____ TO _____	1992 December	91

16 SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the US Government.

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	fusion, Kalman Filter Multisensor Fusion Tracking

19 ABSTRACT (Continue on reverse if necessary and identify by block number) The use of Kalman filtering in tracking targets and the reconstruction of a target's track are addressed in two separate fusion schemes. First, the Kalman filter is used to provide estimates of the position and velocity of a target based upon observations of the target's bearing. Two sensors, a radar in receive mode and an infra-red sensor, take bearings to the target at different scan rates. This information is then fused within the filter to obtain the target's track. Secondly, range, bearing, and frequency are used in fusion. Kalman filtering, Kalman smoothing, and maneuver detection are all used in the reconstruction of a target's track. Improvements are implemented in the method of forcing the excitation matrix and the results documented.

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	UNCLASSIFIED
22a NAME OF RESPONSIBLE INDIVIDUAL TITUS, Hal A.	22b TELEPHONE (Include Area Code) 408-646-2560
	22c OFFICE SYMBOL EC/Ts

Approved for public release; distribution is unlimited.

Application of Kalman Filter on Multisensor Fusion Tracking

by

Brian E. Terpening
Lieutenant, United States Navy
B.A., Louisiana Tech University

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 1992

ABSTRACT

The use of Kalman filtering in tracking targets and the reconstruction of a target's track are addressed in two separate fusion schemes. First, the Kalman filter is used to provide estimates of the position and velocity of a target based upon observations of the target's bearing. Two sensors, a radar in receive mode and an infra-red sensor, take bearings to the target at different scan rates. This information is then fused within the filter to obtain the target's track. Secondly, range, bearing, and frequency are used in fusion. Kalman filtering, Kalman smoothing, and maneuver detection are all used in the reconstruction of a target's track. Improvements are implemented in the method of forcing the excitation matrix and the results documented.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENT

I would like to thank my advisor, Dr. Hal Titus, for his guidance, jokes, and understanding, Very few people can make difficult concepts easy for me to understand. I look forward to our continued friendship. I would also like to express my gratitude to, Dr. Art Schoenstadt, who can not be thanked enough for his generosity, support, and help in refurbishment of all the FORTRAN programs.

I. INTRODUCTION

A. TARGET TRACKING

The detection,location, and tracking of targets plays a critical role in many aspects of military operations. The ability to accurately track and predict the movement of aircraft, anti-ship missiles, and torpedoes in military operations can not be overstated. This report investigates the use of multiple sensors and Kalman filtering in Fusion Tracking.

The Kalman filter has been in use for over three decades in aircraft avionics, radar, sonar, and multiple control problems. This investigation covers a Kalman filter in tracking a target by bearing observations and an additional problem using bearing, range, and frequency observations.

II. THE KALMAN FILTER

The Kalman filter estimates the state of a linear and time-invariant (LTI) system given a set of known inputs to the system and a set of measurements.

A. DEFINITION OF TERMS

The terms used in this chapter and Appendix B, are defined in Table (2.1). Kalman filter system dynamics utilize state space representation, therefore matrix dimensions are given.

TABLE 2.1: Definition of Terms

	Dimension	Code	Symbol
Error covariance matrix:	n x n	P	
Estimate error:	n x 1		
Expected value of error:	n x 1		
Identity matrix:	n x n	eye(n)	I
Kalman gain matrix:		G(k)	G_k
Measurement matrix:	n x 1	H	H
Observation:		theta(k)	z_k
Observation noise:	1 x 1	R	v_k
Residual:	1 x 1		r_k
State estimate:	n x 1	XKK	
State excitation noise:		Q	w_k
State transition matrix:	n x n	Phi	
System state:	n x 1	X	x_k
Transpose:		x'	x'_k

Appendix A defines the terms used in the Kalman filter designed around the torpedo tracking problem. This program is discussed further in Chapter IV.

B. EQUATIONS AND INITIALIZATION

The background given here coincides with that of [Ref 1]. The equations for the filter used in this program are listed below.

$$G_k = P_{k\backslash k-1}H^T[HP_{k\backslash k-1}H^T+R]^{-1} \tag{2.1}$$

$$P_{k\backslash k} = [I-G_kH]\,P_{k\backslash k-1} \tag{2.2}$$

$$P_{k+1\backslash k} = \Phi P_{k\backslash k}\Phi^T+Q_k \tag{2.3}$$

$$\hat{X}_{k\backslash k} = \hat{X}_{k\backslash k-1}+G_k[z_k-\hat{z}_{k\backslash k-1}] \tag{2.4}$$

$$\hat{X}_{k+1\backslash k} = \Phi\hat{X}_{k\backslash k} \tag{2.5}$$

$$\hat{z}_{k\backslash k-1} = H\hat{X}_{k\backslash k-1} \tag{2.6}$$

Terms with a single time subscript (i.e., x_k) refer to the value of the term at that time. Those terms with dual subscripts (i.e., $P_{(k+1\backslash k)}$) refer to the value of the term at a future time given the present states.

The Kalman filter requires initialization and the proper choice of initialization is important. The value of $\hat{X}_{k\backslash k-1}$ is initially set to zero and the following steps are then evaluated in the recursion algorithm.

- Use Equation 2.1 to calculate the Kalman gain.
- Use Equation 2.2 and 2.3 to evaluate the values of the covariance of estimate error matrix for the next iteration.
- Use Equation 2.4 to find the state estimates.
- Use Equation 2.5, given the current state estimates to project ahead for use in the next iteration.

In the program, the bearing theta is simulated and sampled by the filter as if they were taken from a sensor.

III. THE MODEL

In this chapter, we introduce the mathematical and physical model for our first tracking problem. We start by presenting the physical relationship between the target and the observer. After the track has been generated, we use the Kalman filter to verify that the mathematical model is tracking the physical model.

A. PHYSICAL SYSTEMS

The first system used in this investigation consists of a single observer and a single target. For the generation of the target's track, a two-dimensional Cartesian coordinate system with the observer at the origin is used. When referring to Figure 3.1, this x-y grid can be thought of as looking down on a north-south/east-west grid. The target is free to move, unrestricted, throughout the coordinate space while the observer is stationary and remains at the origin.

The information received by the observer consists of bearing to the target. Therefore, the state space estimate appears as:

$$\hat{X} = \begin{bmatrix} \theta \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} \tag{3.1}$$

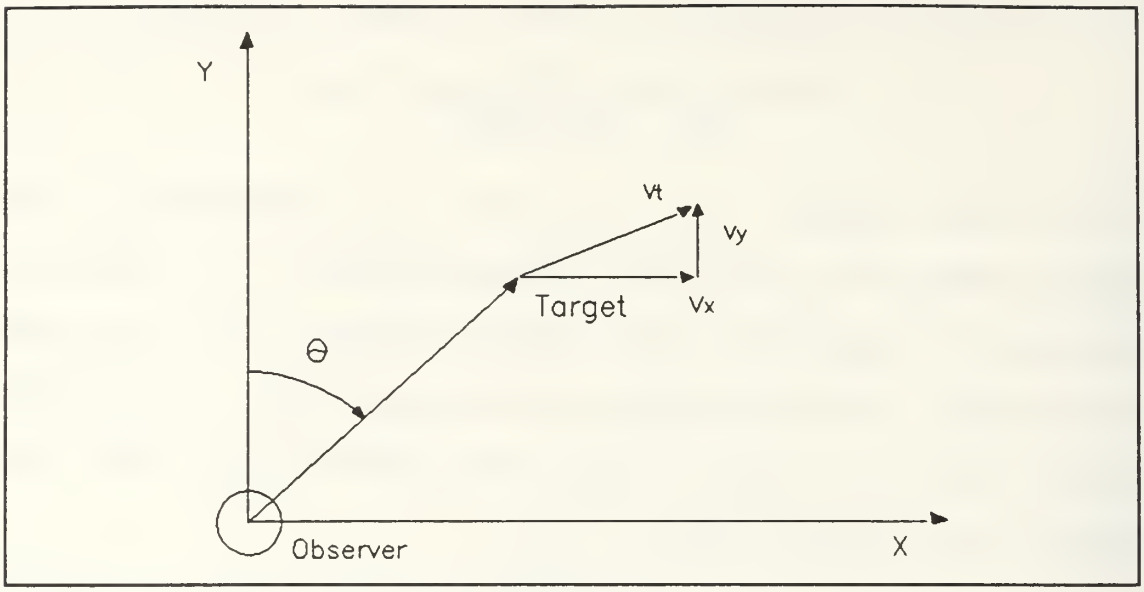

Figure 3.1: First Physical System

These bearings are to be consistent with measurements supplied by a phased array radar, in receive only mode, and most types of infra-red scanners used today. The measurements are taken from the radar 100 times per second, while the infra-red is sampled every second.

B. MODEL TEST

It is critical that the recursive program receive its initial states and observations in the proper order. Therefore, we will examine the Kalman gains for a verification. The MATLAB programs MISSILE1.M and MISSILE2.M used are found in Appendix B. For the purpose of verification, we will assume no noise with the covariance of measurement noise and covariance of excitation equal to zero. If the filter is working properly, the Kalman gains for the

first sample should have the position vector G(1,1) equal to one. The velocity and acceleration vectors, G(2,1) and G(3,1) respectively, should equal zero (see Figures 3.9 through 3.11). At the second sample, G(1,2) is given by $\hat{x}(2) = z(2)$, G(2,2) is given by $(1/T)[z(2)-z(1)]$, and G(3,n) should remain zero. For the third sample, G(3,3) should take on a value given by

$$(1/T^2)\{[z(3)-z(2)]-[z(2)-z(1)]\}. \qquad (3.2)$$

For this simulation the target has v_t = 1 km/sec as shown in Figure 3.3. Figures 3.4 and 3.5 show the distance traveled and change in bearing over one second. The figure showing the change in bearing might be deceptive, as it appears to be a straight line. In fact it is a curved line that reaches its maximum rate of change when the observer is perpendicular to the target's path. Figures 3.6 through 3.8 show the state estimates of the Kalman filter, that gives us a comparison to the actual change in bearing.

Figures 3.9 through 3.11 are the Kalman gains. However, the covariance of measurement noise is 0.001. This avoids a divide by zero in the Kalman gain equations. This will have a minimal effect on the results.

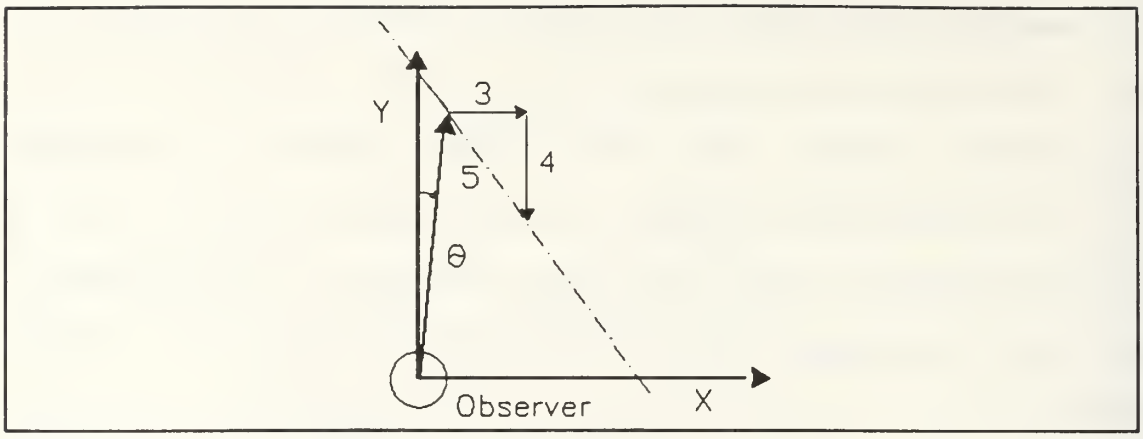

Figure 3.2: Geometry of the target

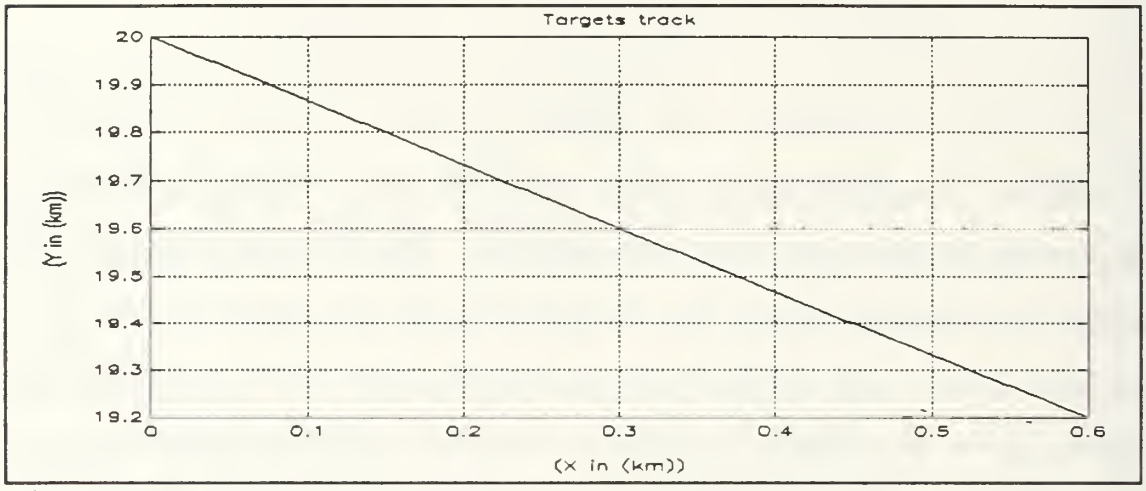

Figure 3.3: Targets track for one second

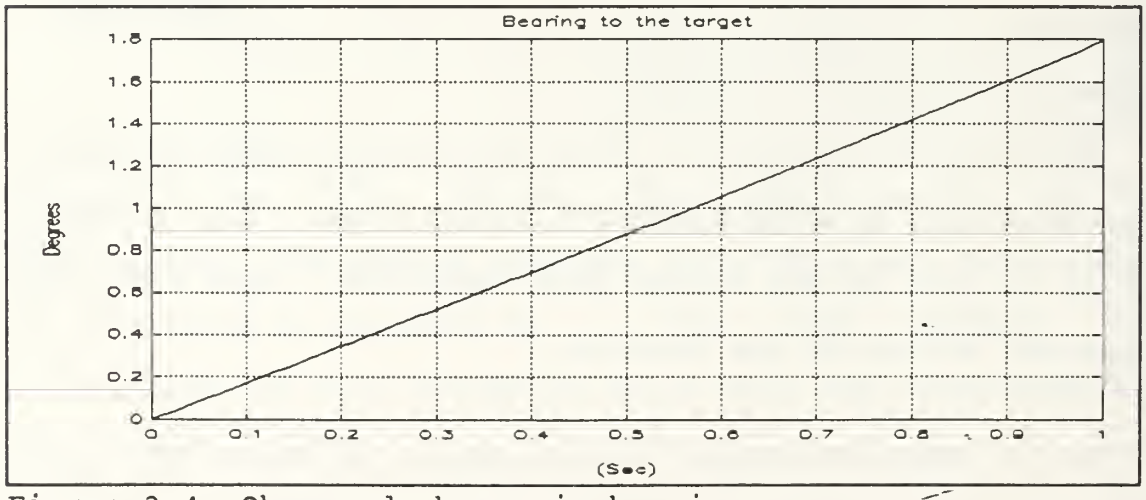

Figure 3.4: Observed change in bearing

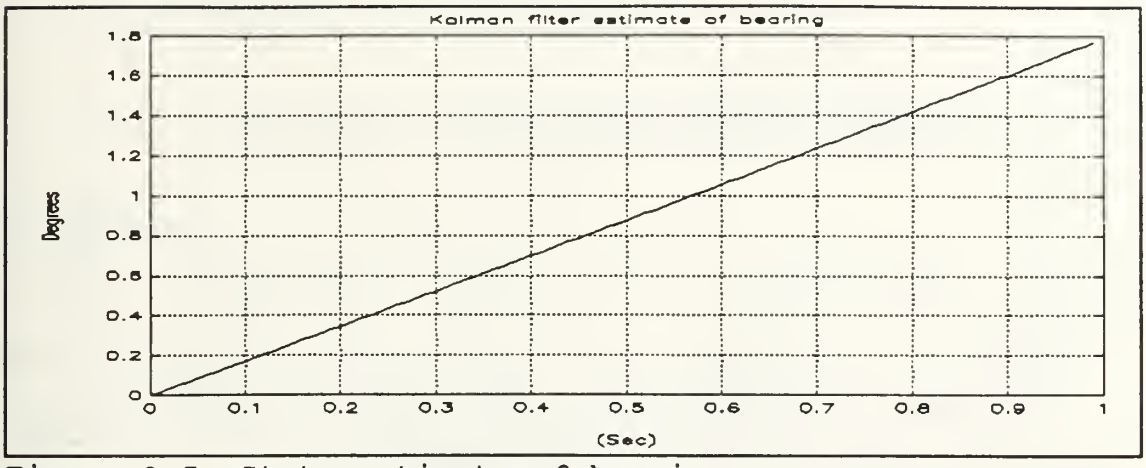

Figure 3.5: State estimate of bearing

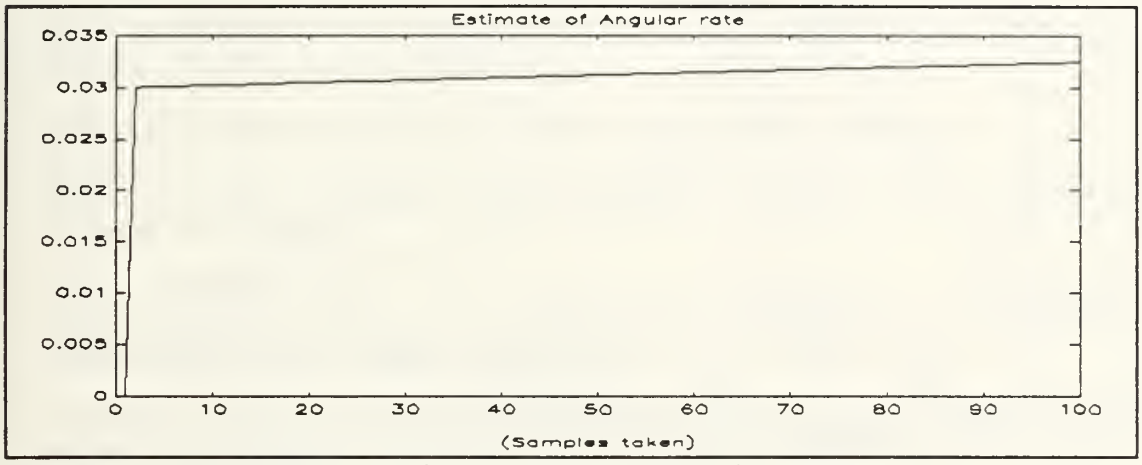

Figure 3.6: State estimate of change in rate

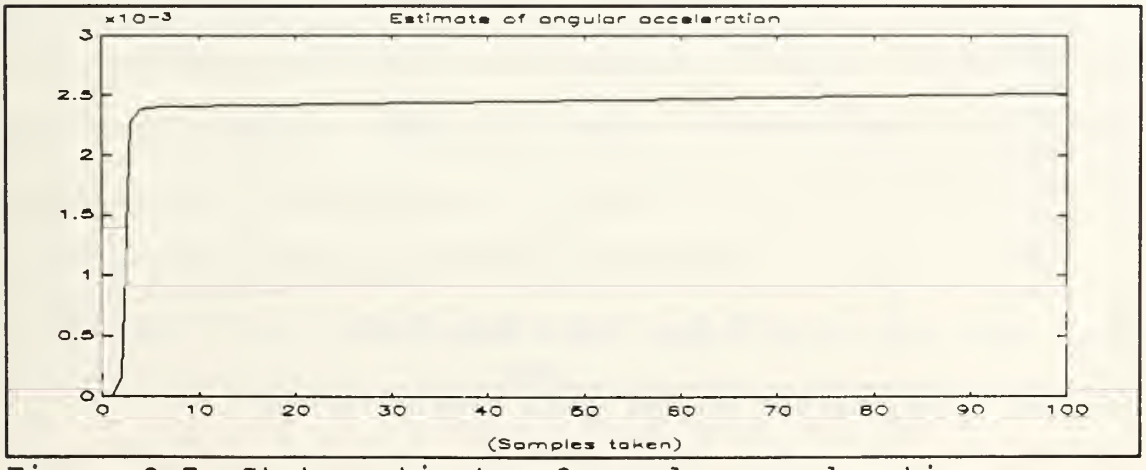

Figure 3.7: State estimate of angular acceleration

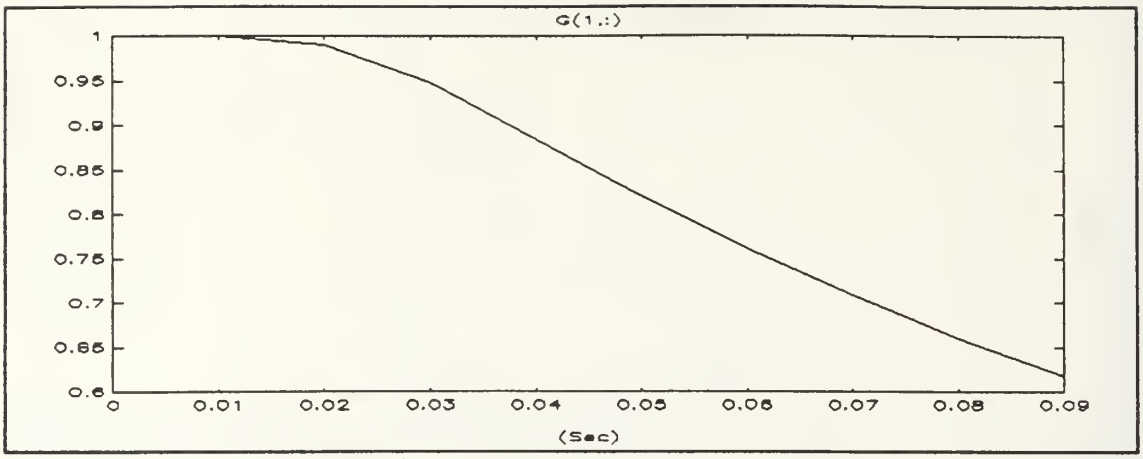

Figure 3.8: Kalman filter gain vector G(1,k)

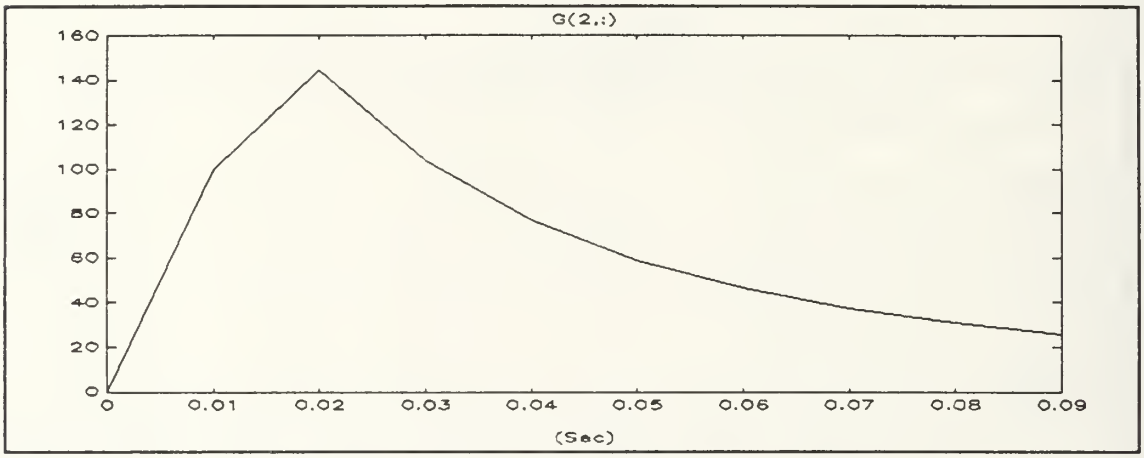

Figure 3.9: Kalman filter gain vector G(2,k)

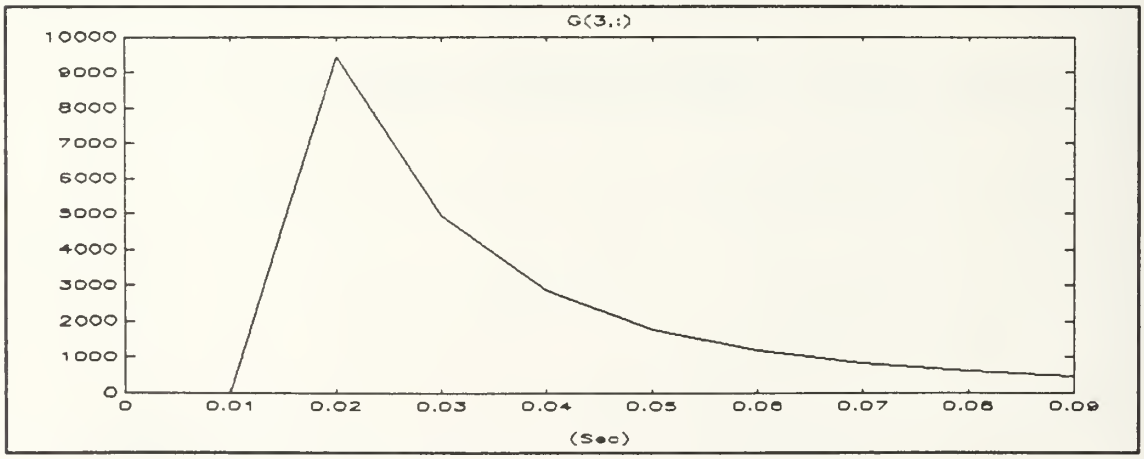

Figure 3.10: Kalman filter gain vector G(3,k)

IV. THE KALMAN FILTER WITH MANEUVER DETECTION

A. PROGRAMS

This chapter is devoted to an explanation of the components and logic used for the programs in Appendix C. TORPMAN.FOR uses the combined work from [Ref 1] and Raymond M. Alfaro. In order to run the program, follow this procedure.

- Run TORPSCUR.M for the s-curve or TRACKSIM.M for the radial track as in [Ref 1].
- GENFILES.M converts the track to a data format.
- POSCONV.FOR converts the MATLAB (ASCII) to FORTRAN (keyport) input.
- Run OBSDATA.FOR to covert position, velocity, acceleration, time, and roll data files to a single observation file OBSERVAT.DAT .
- TORPMAN.FOR is the Kalman filter and may be analyzed by using ANALYZE.M .

B. STATE SPACE AND OBSERVATION MATRICES

As stated previously, this problem utilizes the Cartesian coordinate system. The state space is made up of position, velocity, and acceleration in the x, y, and z vector space, where the dot denotes a time derivative.

$$X = [x\ \dot{x}\ \ddot{x}\ y\ \dot{y}\ \ddot{y}\ z\ \dot{z}\ \ddot{z}]^T \tag{4.1}$$

$$\Phi = \begin{bmatrix} 1 & t & \frac{t^2}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & t & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & t & \frac{t^2}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & t & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & t & \frac{t^2}{2} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & t \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

The observation measurements for our system consists of position and velocity measurements with or without noise added. Observation measurements are defined in the following manner:

$$Z = \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \\ z \\ \dot{z} \end{bmatrix} \tag{4.3}$$

We are assuming the acceleration information is not available. However, if it were possible to get the observation, the filter would take advantage of the opportunity to improve its estimate. We will also assume that some observation may have missing position and velocity data. This will be discussed in depth later in the chapter.

1. Physical Model

For the system as shown in Figure 4.1, the Cartesian coordinate system is used. This implies an array of sensor generate the range measurements. Again the target is free to move throughout the coordinate space. However, the graphic information displays only X-Y coordinates. Frequency information is also utilized in the velocity estimate by estimating the turn counts.

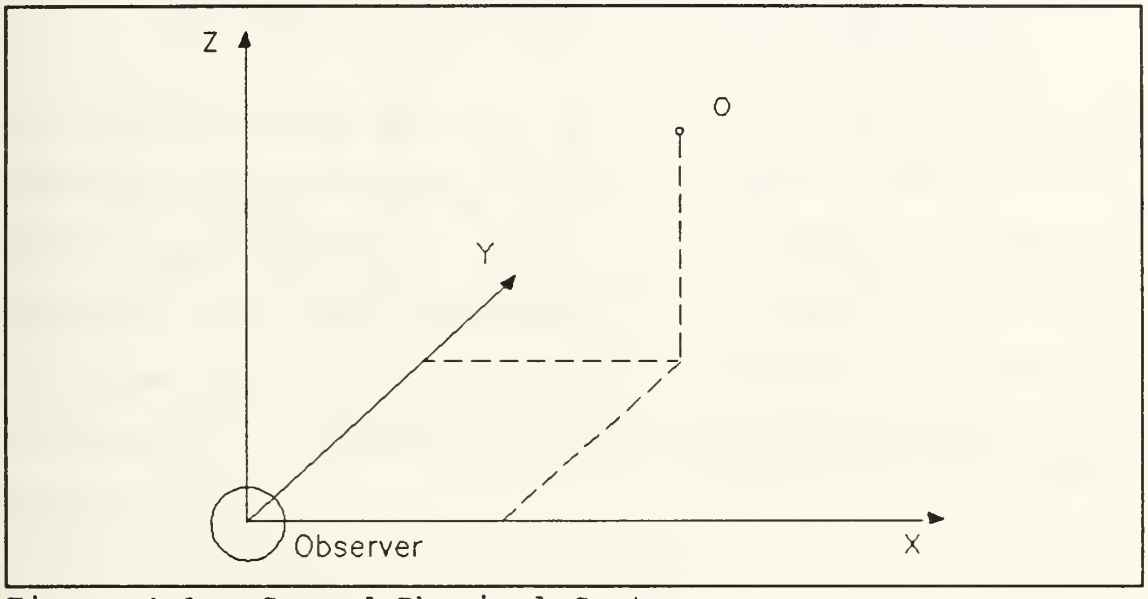

Figure 4.1: Second Physical System

C. WEIGHTED COVARIANCE MATRIX

The covariance of excitation error matrix, Q(k), is considerably more complex. The matrix can be weighted or manipulated when a maneuver is detected. In this new arrangement, the x, y, and z coordinates are adjusted

independently of the others. This matrix was derived in [Ref 2, pp. 36-42] and is given below.

$$Q = \begin{bmatrix} \frac{t^6}{36}w_x & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{t^4}{4}w_x & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & t^2 w_x & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{t^6}{36}w_y & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{t^4}{4}w_y & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & t^2 w_y & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{t^6}{36}w_z & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{t^4}{4}w_z & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & t^2 w_z \end{bmatrix} \quad (4.4)$$

The expression for distance is

$$\frac{t^6}{36}w = s \quad (4.5)$$

where

t = sample time interval,
w = weighting factor, and
s = distance travelled in one sample interval.

A good example of finding w is given in [Ref 1, pp. 22]. One should be reminded, when solving for w, it is the worst case and a smaller weight is more robust.

1. Error Ellipsoids

Error ellipsoids are useful in visualizing the margins of error. The state value should lie within a certain region surrounding the estimate. This uncertainty is expressed in the covariance of error matrix $P_{k\backslash k}$.

The position components submatrix of the error covariance matrix $P_{k\backslash k}$ is defined as

$$P_{xy} = \begin{pmatrix} P_{11} & P_{14} \\ P_{41} & P_{44} \end{pmatrix} = \begin{pmatrix} var\ x & cov(x,y) \\ cov(x,y) & var\ y \end{pmatrix} = \begin{pmatrix} \sigma_x^2 & \sigma_{xy}^2 \\ \sigma_{xy}^2 & \sigma_y^2 \end{pmatrix} \qquad (4.6)$$

The diagonal terms P_{11} and P_{44} of the covariance matrix represents the variances of the estimate in the x and y positions respectfully. If $P_{k\backslash k}$ is a nonnegative definite matrix and the random Gaussian noise processes w(k) and v(k) are independent, a surface of constant probability density can be produced. To elaborate on what is desired, Figure 4.2 is provided.

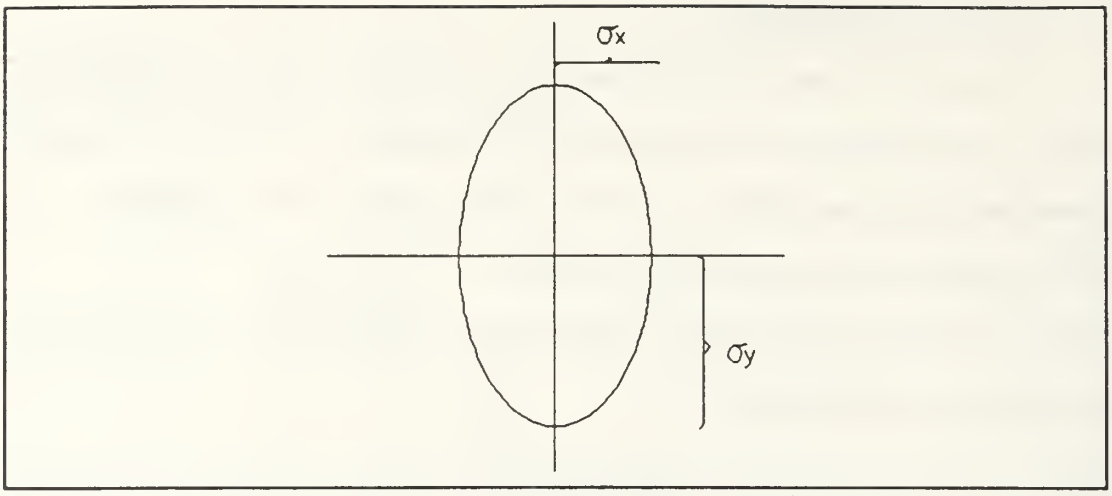

Figure 4.2: Dimensions for the error ellipse

a. Matrix Method

From the density function

$$e^{-x^T P x}, \tag{4.7}$$

we specify the level surface about the origin by

$$x^T P^{-1} x = 1 \quad . \tag{4.8}$$

From numerical analyses [Ref 3, pp. 164], there is a unique upper triangular matrix R such that

$$P = R^T R \tag{4.9}$$

Moreover, this matrix can be stably computed without pivoting. This result, however, is equivalent to stating that there is a corresponding unique triangular matrix $L(=R^T)$ such that

$$P = LL^T \tag{4.10}$$

In terms of the matrix, our basic problem 4.8 becomes

$$x^T (LL^T)^{-1} x = 1 \tag{4.11}$$

However, elementary matrix properties imply that

$$x^T(LL^T)^{-1} = x^T(L^T)^{-1}L^{-1}x$$
$$= (x^T(L^{-1})^T)(L^{-1}x) \qquad (4.12)$$
$$= (L^{-1}x)^T(L^{-1}x) = \|L^{-1}x\|_2^2$$

where $\| \|_2$ represents the usual Euclidean norm. But, this last equality implies that

$$x^TP^{-1}x = 1 \iff \|L^{-1}x\|_2 = 1 \quad . \qquad (4.13)$$

Therefore, the basic problem 4.10 is equivalent to

$$\|L^{-1}x\|_2 = 1 \quad . \qquad (4.14)$$

This equation is solvable if and only if,

$$L^{-1}x = y \text{ for some vector } y \text{ such that } \|y\|_2 = 1 \qquad (4.15)$$

But,

$$L^{-1}x = y \Leftrightarrow x = Ly \quad . \qquad (4.16)$$

Therefore, we conclude that the desired error ellipsoid is specified by

$$\{ x | x=Ly \ , \ \|y\|_2=1 \} \qquad (4.17)$$

b. Reaction to a maneuver

The effects of maneuver detection are seen in Figure 4.3. As the filter detects an acceleration in one direction, the uncertainty of the position increases in that dimension. The ellipsoids can be used to verify the filters ability to predict the movements of the target. The program ELLIPSE.M that generates the ellipsoids can be found in Appendix B.

c. Initialization check

The program S_ELLIPSE.M contained in Appendix C give us the same visual verification of the filter using the

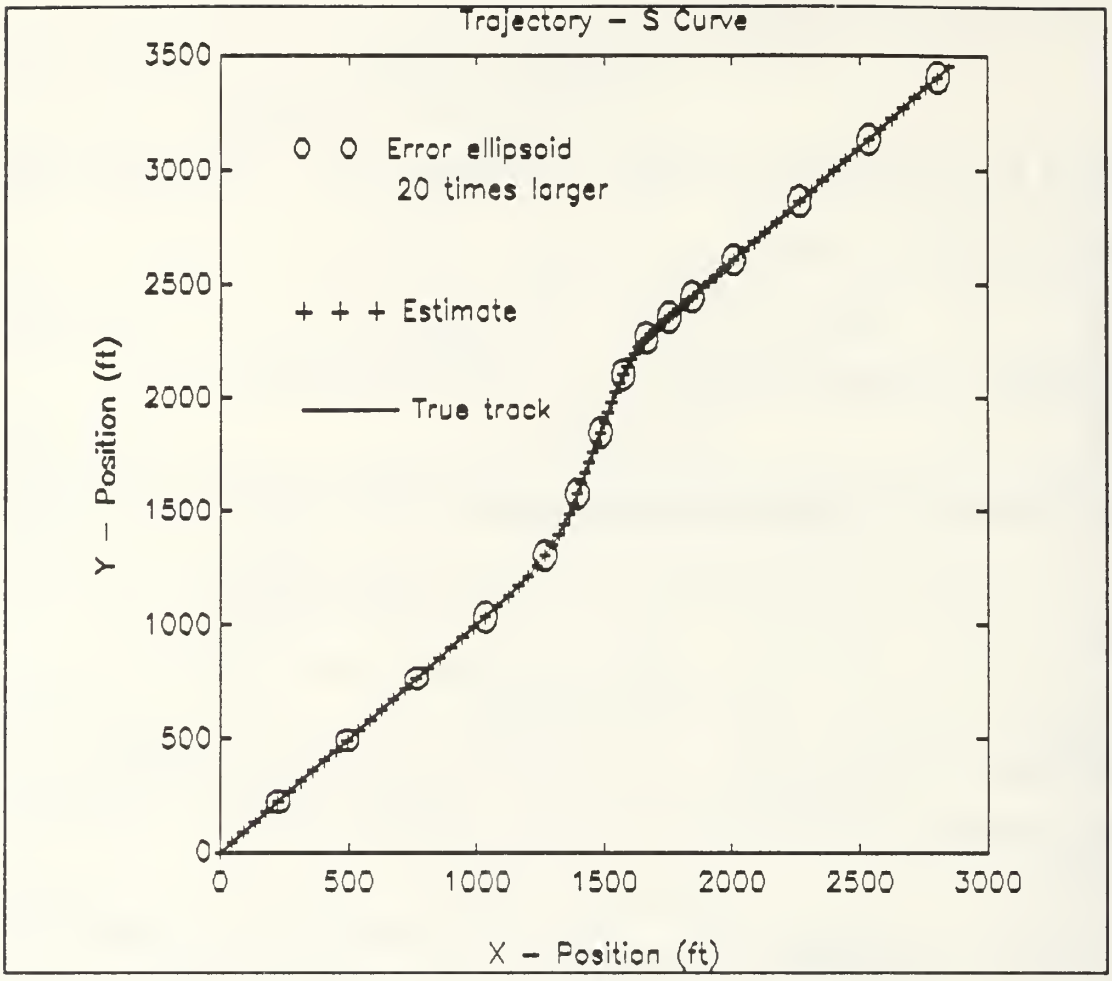

Figure 4.3: The effect of maneuver detection on the covariance of error

same idea. By using the error covariance matrix P we can obtain a visual insight into the performance of the filter. Figure 4.4 shows the exponential decay of the probability of error.

D. THE OBSERVATION MATRIX - H

The logic for the observation matrix starts with the manipulation of data in OBSDATA.FOR, in Appendix C. After receiving an observation, the data is manipulated to obtain a

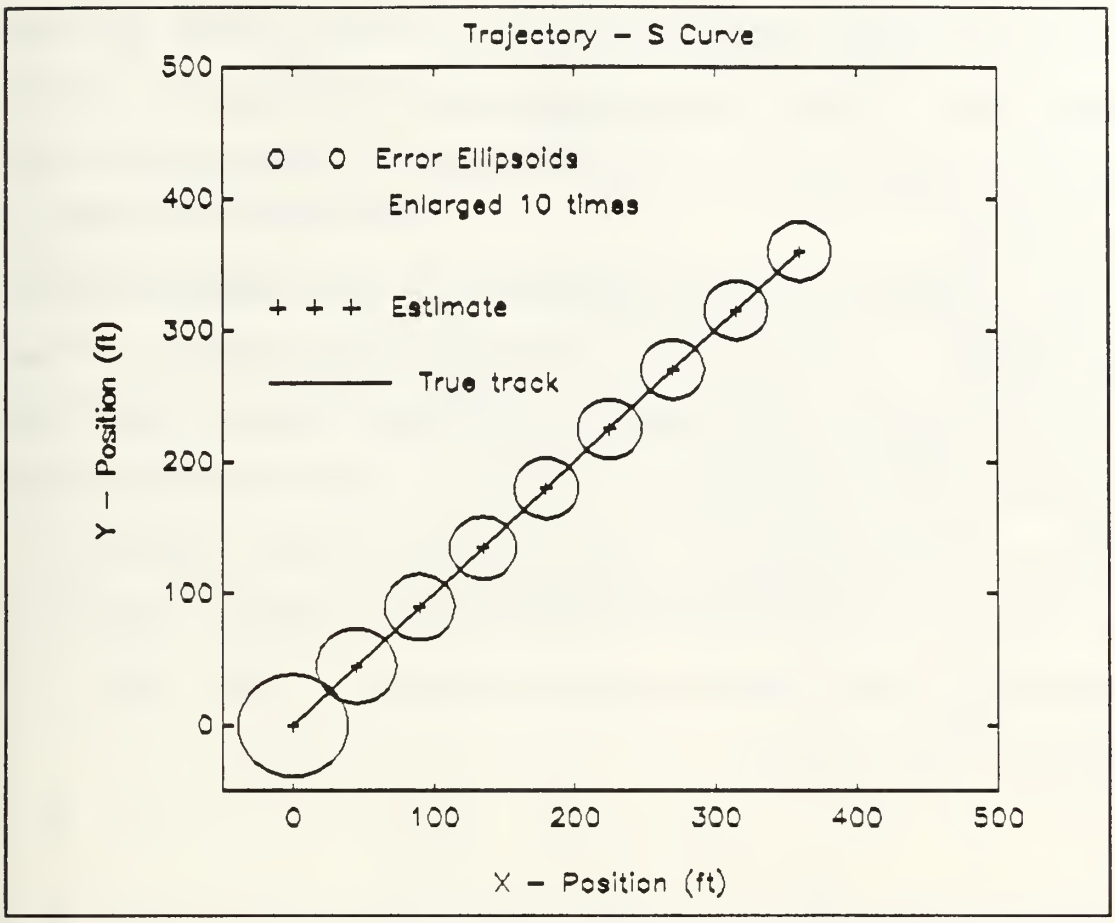

Figure 4.4: The correct response for the covariance of error ellipsoids

state space vector with position, velocity, and acceleration. Based on which state variables are actually present in the observation, a unique binary signal is generated. Take the coefficient of the power

$$2^{i-1} = \begin{cases} 1 \text{ if } x_i \text{ is present in the observation} \\ 0 \text{ if } x_i \text{ is not present} \end{cases}$$

The number is stored with the position vector (x) as low bit. For example:

SRC = 11 =

0	0	0	0	0	1	0	1	1

The binary number implies that x, $\dot{x}$ and y make up the observation. This allows subsequent decoding of which components are present by a combination of divide by 2 and modulo opus.

The matrix H, can be produced for any combination of measurements. Given the same statistical characteristics as the actual data, it is possible to have missing data and multiple returns from adjacent arrays. This filter has the ability to continue when pieces of data are not received, or when the two sensors are operating at slightly different frequencies. The scheme used here helps in this regard in that it is more robust.

V. RESULTS

A. BASELINE INFORMATION

The s-curve data is run through the filter from [Ref 1, pp. 61]. This information will now be used in determining improvements in performance. The knowledge that the filter performed best with the excitation matrix weighted to 1500, is utilized in the production of Figure 5.1. The

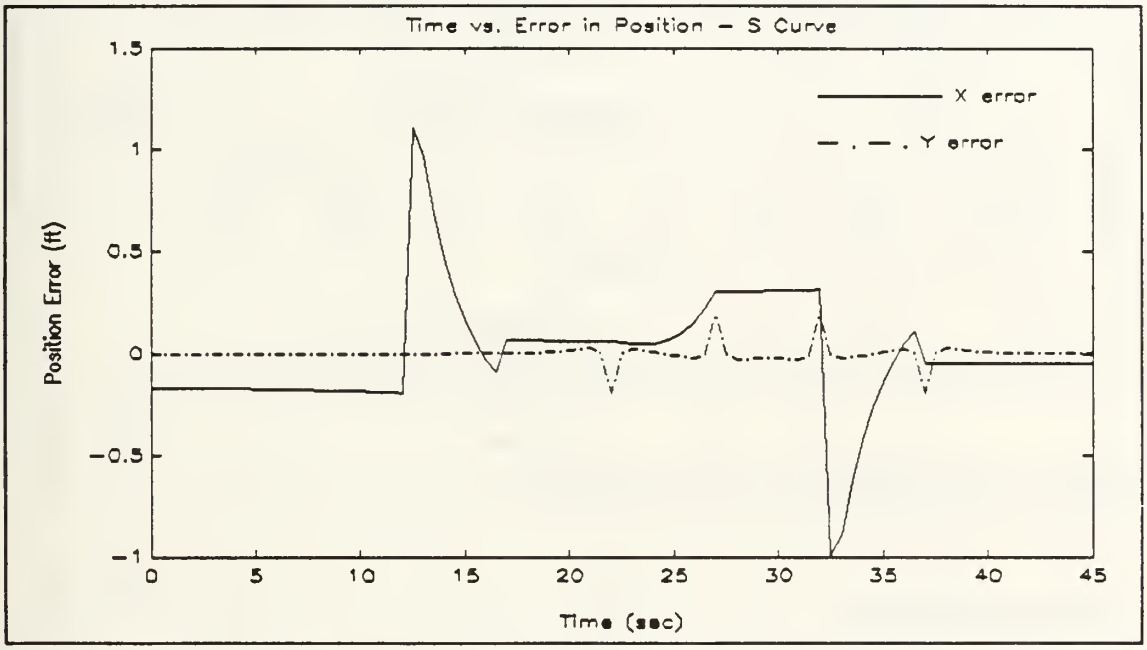

Figure 5.1: X and Y smoothed estimate error with old maneuver detection scheme

performance of the filter can be improved in this noiseless environment by increasing the figure for the weighing from 1500. However, as shown in [Ref 1], the filter's performance in a noisy environment would be degraded.

For a better idea of what is causing the errors, Figure 5.2 shows when the target is maneuvering. Looking at where the maneuvers occur, coincides with the disturbance in tracking.

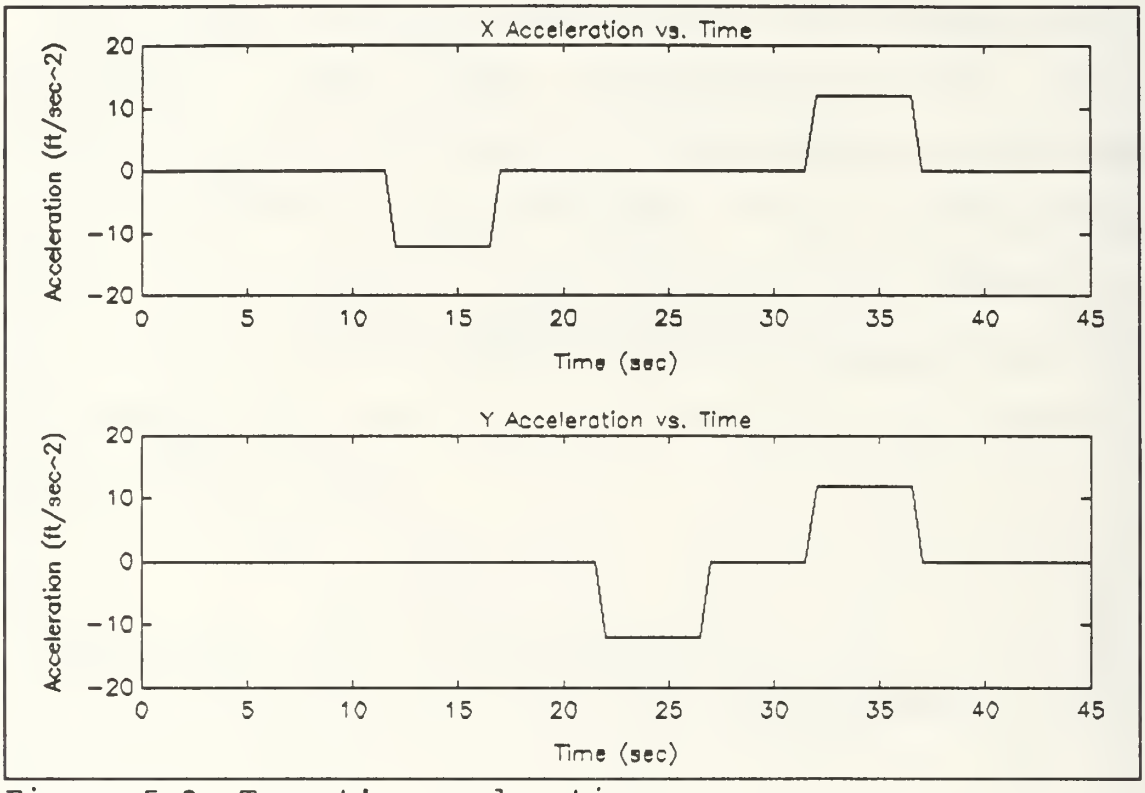

Figure 5.2: Target's acceleration

B. MODIFICATIONS

The filter's maneuver detection scheme and observation measurement matrix, (H), are now as discussed in the previous chapter. This filter is modified in one other respect. This filter manipulates the excitation matrix when a maneuver is detected, not after the maneuver is detected. The purpose of estimating the acceleration is to be able to sense the

maneuver simultaneously with the onset of the maneuver. "None of the time lag associated with the residual method is experienced." [Ref 1] The results are readily seen in Figure 5.3. Again, this is with no noise and the Q matrix is weighted at unity. A view of the targets path is provided in Figure 5.4.

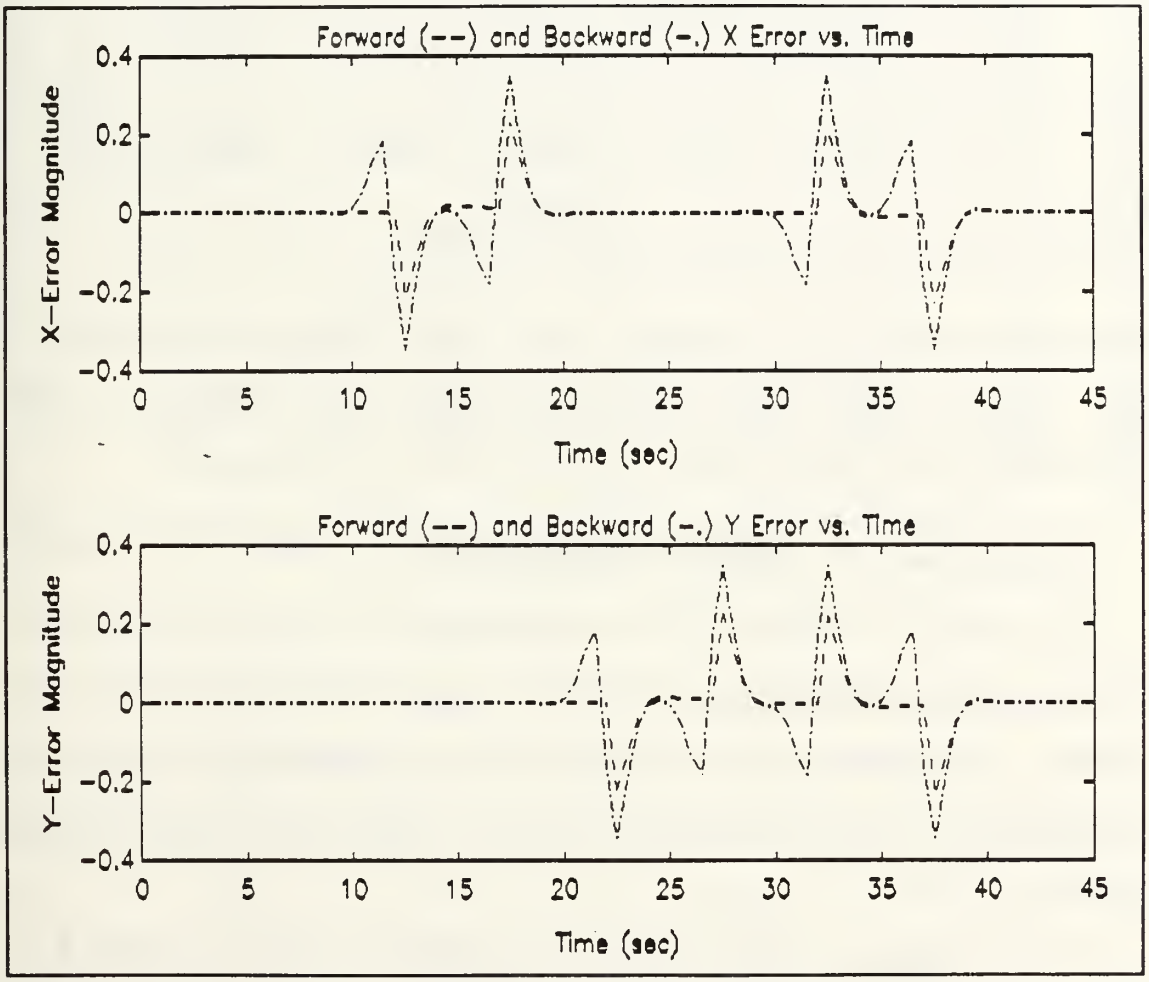

Figure 5.3: X and Y error for the s-curve using the new filter; weight is set at unity

Note that the smoothing in backward filter causes the larger of the errors in a noiseless environment. The more the

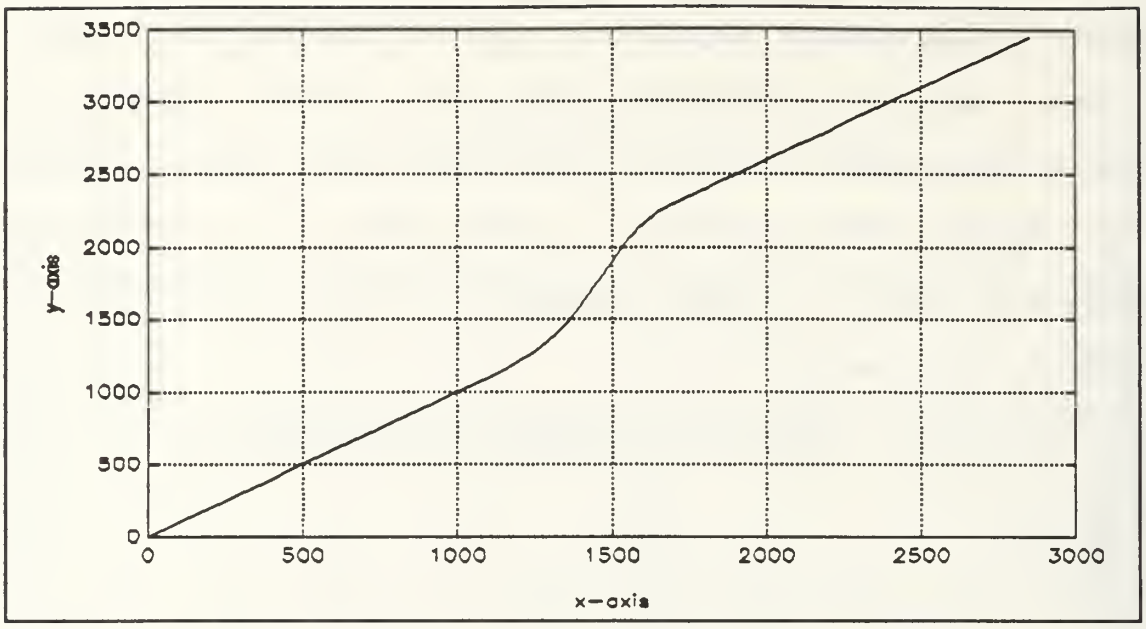

Figure 5.4: The targets track in feet

excitation matrix is weighted the less the errors become. But, again, this does not hold true with noise in the environment and the trade offs need to be evaluated.

C. EVALUATION FOR OPTIMAL PERFORMANCE

In order to evaluate the differences in the two filters, we must first establish the optimal performance for this new filter. Multiple tracks are created to establish a value for the weight and to gather enough data to reduce any statistical errors.

As discussed in [Ref 1, pp. 31] Gaussian noise is added to the track with a distribution variances of 15 square feet, 0.01 square yards/second, and 0.03 square feet/second squared for position, velocity, and acceleration respectively, with

zero mean. A figure of such a track is provided in Figure 5.5.

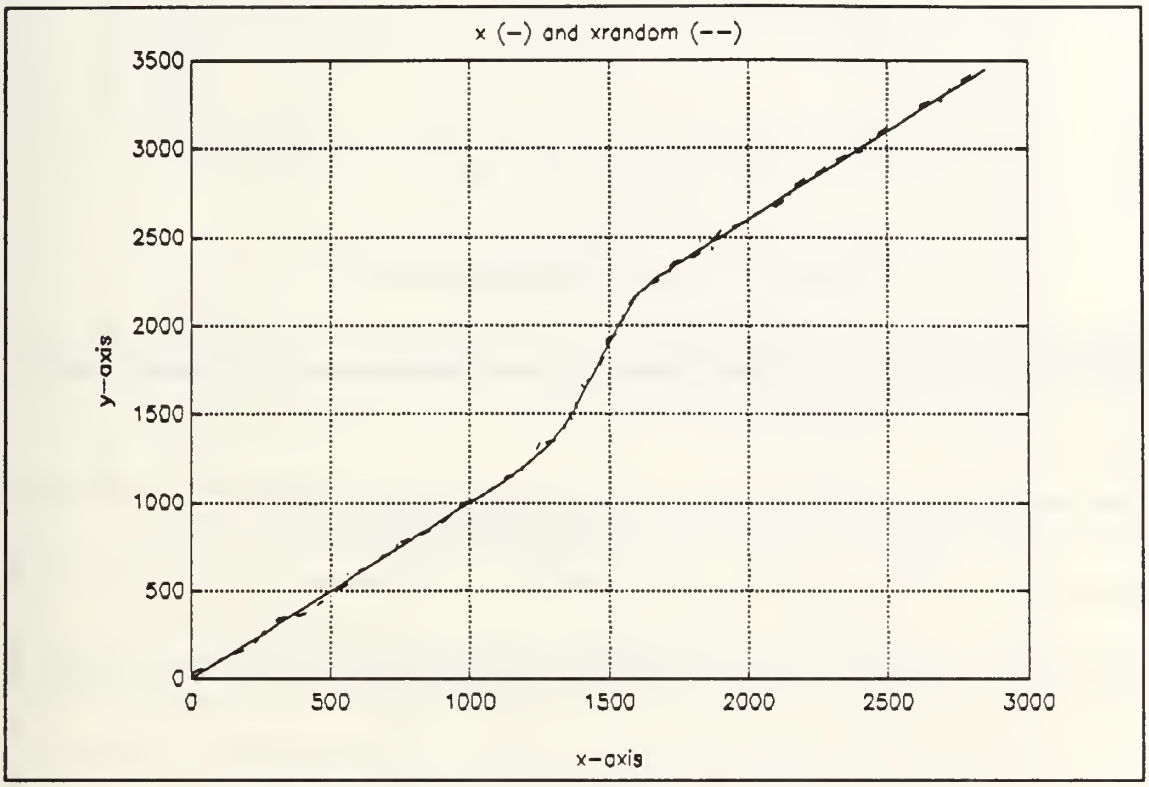

Figure 5.5: Target track with noise added

Trial and error is used to find the optimal performance of the filter. The numbers seen in Table 5.1 are the mean radial distance errors obtained from the program in [Ref 1, pp. 100]. Table 5.1 shows a few of the multiple iterations of combinations used in choosing the optimal weights. The maximum weight (WTMAX) and minimum weight (WTMIN) are chosen to be 100 and 0.1 respectively. This choice offers a half foot reduction in mean radial distance error in a relatively quiet environment, while giving up a little less than a half

foot in this noise filled environment. Figure 5.6 shows how the filter handles a noisy track for one run.

TABLE 5.1: Table of weights

		Maximum Weight					
		500	100	10	1	0.1	0.01
M I N	100	4.27	4.03	--	--	--	--
	10	3.18	2.86	2.3	--	--	--
	1	3	2.67	2.3	2.23	--	--
	0.1	2.96	2.66	2.28	2.20	2.22	--
	.01	2.98	2.67	2.28	2.22	2.31	3.36

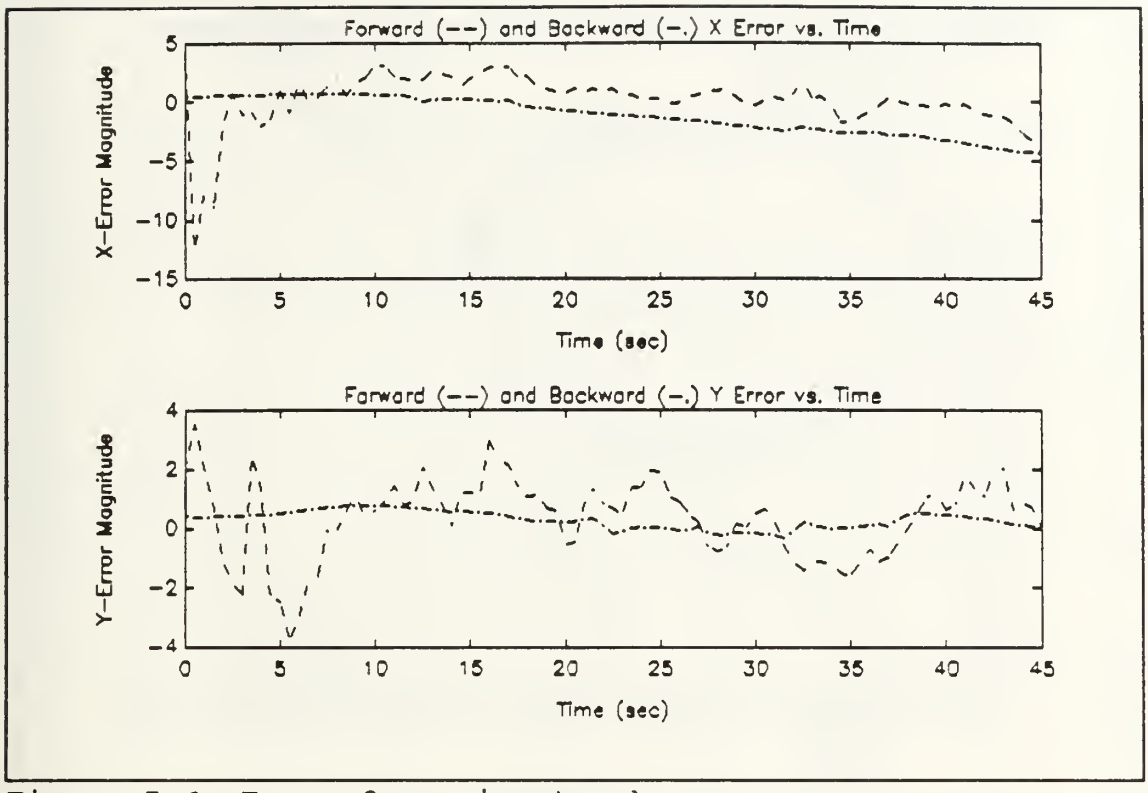

Figure 5.6: Error for noisy track

D. COMPARISONS

Both filters, using all of their best features, are used on a track with approximately 20 feet of mean radial distance error. Figure 5.7 provides a visual comparison as to the capability of each filter in the same figure. The mean radial distance errors in the upper left hand corner show the difference in performances of the old and new filters.

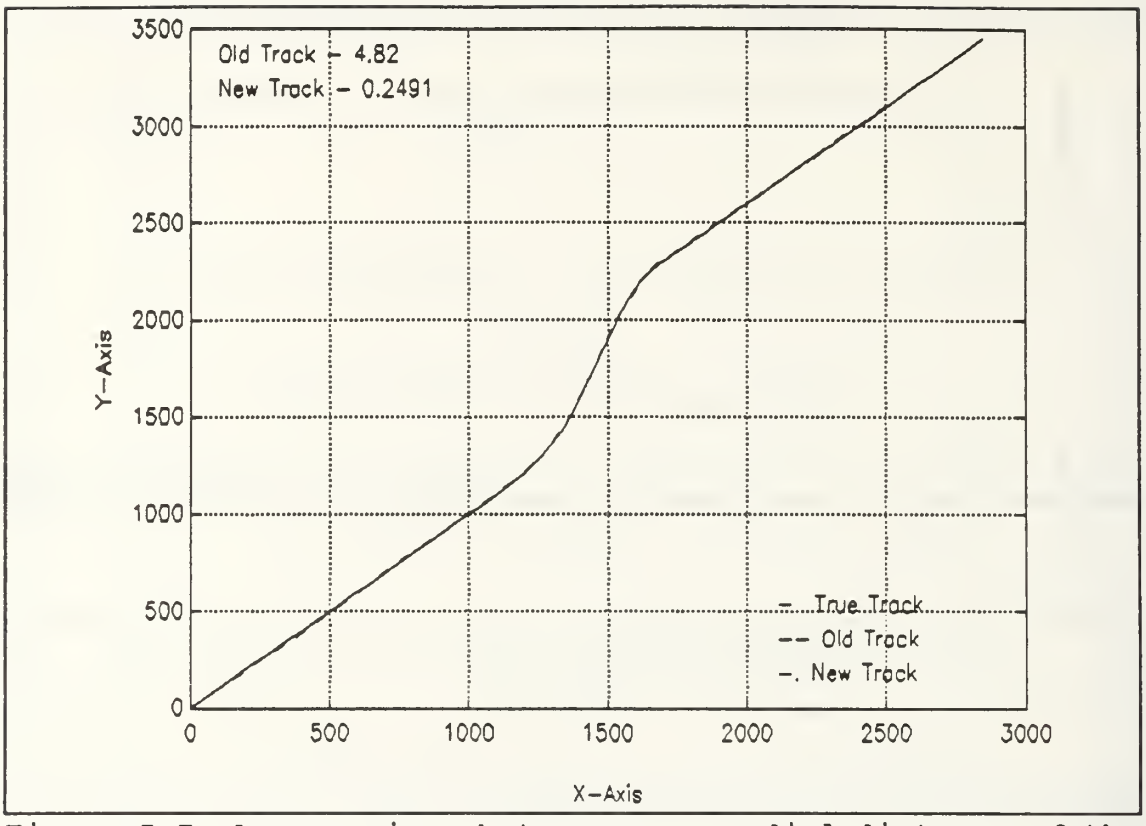

Figure 5.7: A comparison between mean radial distance of the old and new filter

For the circular maneuver presented in [Ref 1,pp. 36], we will look at the best result for the old filter with optimal settings. In this track the torpedo makes 360 and 270 plus degree turns. The old mean radial distance error was 5.951 . In Figure 5.8, the mean radial distance errors are in the upper left hand corner. The old track is the "raw" observation while the new track is the smoothed filter estimate of the true track. The mean radial distance error show over a 50% improvement with an error of 2.165 .

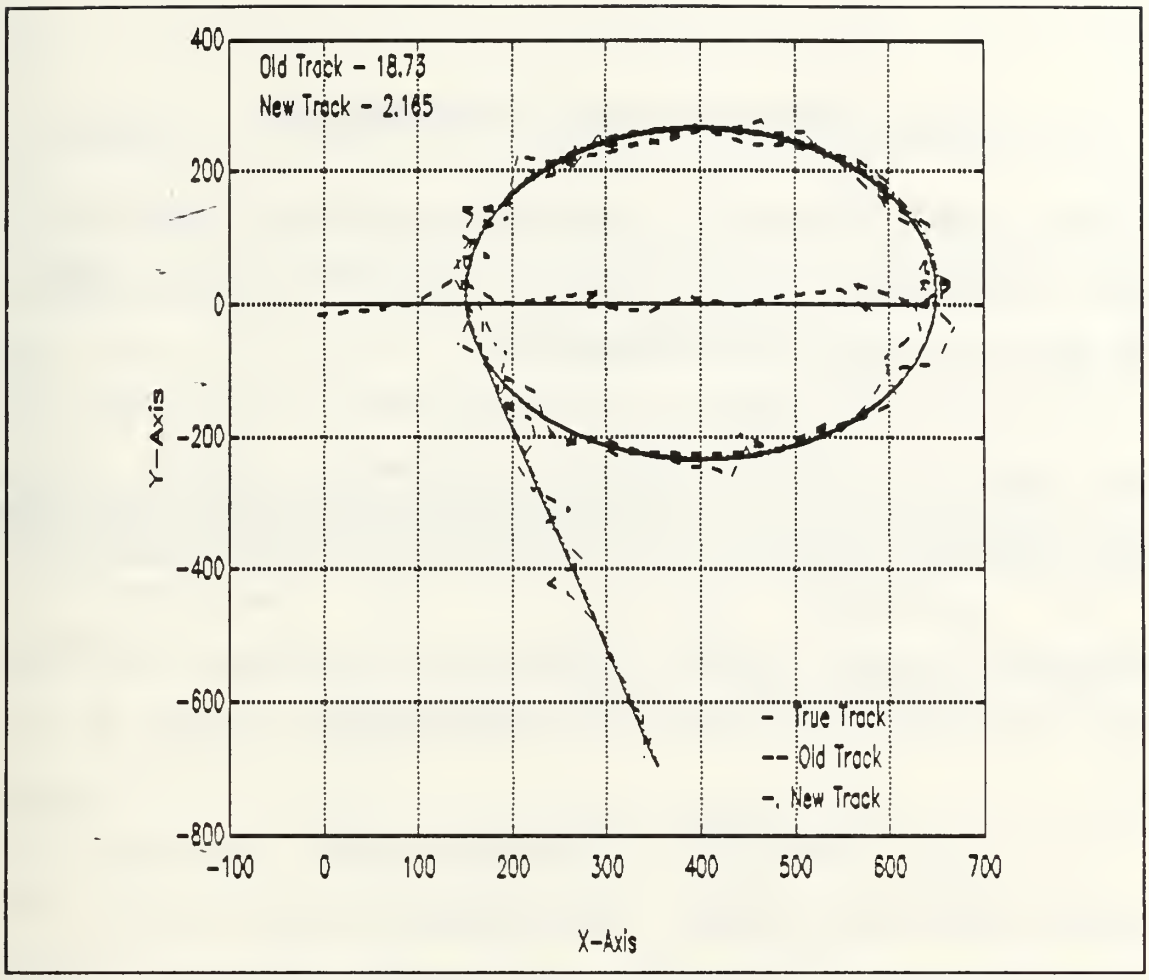

Figure 5.8: Filter output with WTMIN=0.1 and WTMAX=100.0

VI. CONCLUSIONS AND RECOMMENDATIONS

The objective was to investigate uses of fusion in tracking problems. In the process, there were several items that bore out special investigation.

The program for torpedo track reconstruction now works in three dimensions and its overall performance improved. One major improvement was the creation of the OBSDATA.FOR program and its subsequent improvement on the observation or measurement matrix. This revamping of the program and its logic has improved its overall characteristic another 50 to 80%.

The tracking of targets by electromagnetic and infra-red sensors needs to be pursued. The sensors in [Ref 4], all have the capability of tracking fast moving targets. The preliminary results look good for three polar coordinate filters, all working in parallel, to track air traffic passively. If there were one filter per sensor, and one filter to fuse and triangulate bearings using a conversion vector like in [Ref 4, pp.4-4], then vessels would be able to track traffic passively in conditions of restricted electromagnetic emissions.

APPENDIX A: DEFINITION OF TERMS FOR USE IN TORPMAN.FOR

The terms and characters defined here are for easy access to those trying to comprehend the program in Appendix C. The logic for the program is discussed in more depth in Chapter IV.

TABLE A.1: Terms used in TORPMAN.FOR

$3DTEMP	This is a direct access file for storing values of the error covariance matrix ($P_{k\backslash k}$). The character associated with this file is PKKS3D.
$M1TEMP	This is a direct access file for storing values of PKKM1S or the smoothed $P_{k\backslash k-1}$.
ADD	A subroutine which adds two input matrices.
ACCTHRSH	The acceleration threshold is initialized to 5 feet/second. This is for use in the maneuver detection program, MANDET, and may be changed within the SETTHRSH subroutine.
CHANGE	This is a subroutine which allows the user to change the weight (WTMIN or WTMAX) and error covariance matrix at time k minis 1 (PKKM1). The program is therefore altered in behavior without having to compile.
H	The measurement matrix is discussed in more depth in chapter V.
NDIMEN	The number of dimensions used are 1, 2, or 3 D.
NUMSTA	This is the number of states in the vector space.
NSTMAT	This is used in FORTRAN memory allocation for one time step.

NOBVEC	This allocates an array large enough for the observed data set.
OBSERVAT.DAT	The observed data base.
PHIDEL	This subroutine builds the Phi and Del matrix.
PKK	The error covariance matrix ($P_{k\backslash k}$).
PKKM1	The error covariance matrix ($P_{k\backslash k-1}$), which is at time minus one.
PKKM1S	This is the smoothed PKKM1 from the backward Kalman filter.
PKKOUT.DAT	An output file is created containing the information for error ellipse in the x and y plane.
Q	State excitation noise matrix.
R	Observation noise matrix.
ROLTHRSH	The roll threshold is initialized to 5 degrees/second. This can be changed with the SETTHRSH subroutine.
SCR	An integer, when read as a binary number, reveals the make up of the observation.
TMPV1 & 2	A temporary storage space for characters as they change value from k/k-1 to k/k to k+1/k = k/k-1 .
TEMP1, 2, & 3	A temporary storage space for characters as they change value from k/k-1 to k/k to k+1/k = k/k-1 .
WTMAX	This is the maximum weight used upon the detection of a maneuver. The value is preset to 20,000 but can be altered in the CHANGE subroutine.
WTMIN	This is the minimum weight used in the program. Which insures that the Kalman gains do not approach zero if the target is not maneuvering.
XKK	The state estimate ($\hat{x}_{k\backslash k}$).

XKKM1	The state estimate ($\hat{x}_{k\backslash k-1}$), which is at time minis one.
XKKFWD.DAT	The state estimates of the Kalman filter are stored in this data file.
XKKS	Observations are stored in this matrix, including acceleration obtained from ΔV. It is then used for the state estimate and smoothed state estimate as it passes through forward and backward Kalman filter.
XKSMOOTH.DAT	The state estimates of backward filtering (XKKS) are stored in this file.
ZZ	Stores the observations.

APPENDIX B: MATLAB PROGRAMS

A. ANALYZE.M

```
%analyze.m
%  This  M-file compares the output of a Kalman Filter Run
%  with the true x, y, and z positions.
clg
!del data\six.met
! cd data
load tracknum.dat
tracknum = fix(tracknum) ;
load trueposn.
load xkkfwd.dat
load xksmooth.dat
%
errfwd=xkkfwd(:,2:4)-trueposn ;
errbwd=xksmooth(:,2:4)-trueposn ;
%
eval( [ '! del filtr',num2str(tracknum),'.met' ])
subplot(211) ;
plot(xkkfwd(:,1),errfwd(:,1),'--g',..
        xkkfwd(:,1), errbwd(:,1),'-.w' )
xlabel('Time (sec)');ylabel('X-Error Magnitude' );
title('Forward (--) and Backward (-.) X Error vs. Time');
%
plot(xkkfwd(:,1),errfwd(:,2),'--g',..
         xkkfwd(:,1),errbwd(:,2),'-.w')
xlabel('Time (sec)');ylabel('Y-Error Magnitude' );
title('Forward (--) and Backward (-.) Y Error vs. Time');
eval( [ 'meta filtr',num2str(tracknum) ])
meta six
pause
%
clg
subplot(211)
plot(xkkfwd(:,1),errfwd(:,3),'--g',..
         xkkfwd(:,1),errbwd(:,3),'-.w')
xlabel('Time (sec)');ylabel('Z-Error Magnitude' );
title('Forward (--) and Backward (-.) Z Error vs. Time');
%meta
pause
clg
%
eval('! cd ..')
```

B. ELLIPSE.M

This program will plot an error ellipsoid once ever six observations.

```
!del pic2.met
load c:\matlab\wiseman\output2.dat
load c:\matlab\wiseman\truepos2.dat
load c:\matlab\wiseman\pkkout.dat
axis('square')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% plot path and estimate of path                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
plot(output2(:,1),output2(:,2),'+',
                              truepos2(:,2),truepos2(:,3))
xlabel('X - Position (ft)');ylabel('Y - Position (ft)')
title('Trajectory - S Curve')
gtext('____ True track')
gtext('+ + + Estimate')
gtext('O   Error ellipsoid')
gtext('20 times larger')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% generate a uniformly spaced set of points                                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
delt=-1:.05:1;
delt=pi*delt;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% generate a (2xm) set of vectors on the unit circle                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
y=[cos(delt);sin(delt)];
shift=diag(ones(41));
shift=shift';
hold on
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The function chol, to compute the Choleski                                    %
% decomposition, is done "backwards" in MATLAB. It                              %
% makes an upper triangular matrix R, not L. So take                            %
% the transpose.                                                                %
% An array of x-vectors (2xm) has been created that                             %
% form the error ellipsoid.                                                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for k=6:6:max(size(output2)),
   P=pkkout(k*2-1:k*2,1:2);
   s=chol(P)';
   x=s*y;
   plot(x(1,:)*20+output2(k,1)*shift,
                      x(2,:)*20+output2(k,2)*shift)
end
hold off
axis('normal')
meta pic2
```

C. GENFILES.M

```
% THIS PROGRAM CONVERTS SIMULATED TORPEDO TRACKS GENERATED BY
% THE MATLAB PROGRAM  TORPGEN.M  INTO FILES SUITABLE FOR INPUT
% TO THE FORTRAN PREPROCESSOR  OBSDATA.FOR.
% AFTER CONVERSION INTO STANDARD INPUT FORMAT BY OBSDATA.FOR,
% THE DATA WILL BE TESTED AGAINST THE KALMAN FILTERING PROGRAM
% TORPMAN.FOR, WHICH IS DESIGNED TO RECONSTRUCT TORPEDO TRACKS
% GENERATED AT THE UNDERSEA RANGE AT KEYPORT, WA.
%
%
%
%%%%%%%  Prompt User for Desired Type and Run   %%%%%%%%%%%%%%
%%%%%%%       Number for Data File              %%%%%%%%%%%%%%
%
! cls
disp(' Select Type of Simulated Track to Process ' )
disp(' ')
disp('    1.  True Track (w/o pseudo-random noise added)' )
disp('    2.  Track with Pseudo-Random Noise Added' )
disp(' (Any other value will abort processing)')
disp('  ')
ntrktyp  = input('        Selection: ') ;
if ntrktyp <= 0 ; return ; end
if ntrktyp >  2 ; return ; end
%
disp('  ');
trutrk = input(['Enter the Sequence Number of the True Track
               File:      ']) ;
obsfil = ['true'];
obstrk = trutrk;
if ntrktyp == 2
   disp('  ');
   obsfil = ['tst'];
   obstrk = input(['Enter the Sequence Number of the Simulated
                  Track File: ']) ;
end
%
%%%%%%%%%%  Input the Desired Files and Produce    %%%%%%%%%%
%%%%%%%%%%  Standard Output Files That Will Run    %%%%%%%%%%
%%%%%%%%%%       with Program POSCONV.FOR          %%%%%%%%%%
%      trueposf = ['truep',int2str(trutrk) ] ;
       deltatf  = ['delt',int2str(trutrk) ] ;
       eval( [ 'load data\',trueposf   ])
       eval( [ 'load data\',deltatf    ])
%
       save  data\trueposn x       /ascii
       save  data\deltsimu deltat  /ascii
%
       positionf = [ obsfil,'p',int2str(obstrk) ] ;
       velocityf = [ obsfil,'v',int2str(obstrk) ] ;
```

```
        accelerаf = [ obsfil,'a',int2str(obstrk) ] ;
        eval( [ 'load data\',positionf  ])
        eval( [ 'load data\',velocityf  ])
        eval( [ 'load data\',accelerаf  ])
%
        save  data\tracknum.dat   obstrk   /ascii
        save  data\simulpos   x     /ascii
        save  data\simulvel   v     /ascii
        save  data\simulacc   a     /ascii
%
^Z
```

D. MISSILE1.M

```
MISSILE1.M
This program was developed for tracking a target moving at
approximately mock three.  The problem is set up for the poler
coordinate system.

dt1=0.01;
kmax=100;

%Missile
speed=1;                    %approx. 3.3 time the speed of sound
xrange=0;                   %km
yrange=20;                  %km
x=zeros(4,kmax);
x(:,1)=[xrange 3*speed/5 yrange -4*speed/5]';
theta=zeros(1,kmax);
theta(1,1)=atan2(x(1,1),x(3,1));
time=zeros(1,kmax);
A=[0 1 0 0;0 0 0 0;0 0 0 1;0 0 0 0];
B=[0 0;1 0;0 0;0 1];
[phi,del]=c2d(A,B,dt1);

%Filter
R=0.001;
Q=0;
A=[0 1 0;0 0 1;0 0 0];
B=[0 0 1]';
H=[1 0 0];
[Phi,Del]=c2d(A,B,dt1);
G=zeros(3,kmax);
XKK=zeros(3,kmax);
XKKM1=zeros(3,kmax);
PKKM1=eye(3)*1e7;
for k=1:kmax
   G(:,k)=PKKM1*H'*(H*PKKM1*H'+R)^(-1);
   PKK=(eye(3)-G(:,k)*H)*PKKM1;
   PKKM1=Phi*PKK*Phi'+Q;
   XKK(:,k)=XKKM1(:,k)+G(:,k)*(theta(k)-XKKM1(1,k));
   XKKM1(:,k+1)=Phi*XKK(:,k);
   x(:,k+1)=phi*x(:,k);
   theta(k+1)=atan2(x(1,k+1),x(3,k+1));
   time(k+1)=time(k)+dt1;
end
plot(x(1,:),x(3,:)),grid,ylabel('(Y in (km))')
title('Targets track'),xlabel('(X in (km))')
meta test1
plot(time,theta*180/pi),grid,title('Bearing to the target')
xlabel('(Sec)'),ylabel('Degrees')
```

```
meta test2
plot(time(1,1:kmax),XKK(1,:)*180/pi),grid
title('Kalman filter estimate of bearing')
xlabel('(Sec)'),ylabel('Degrees')
meta test3
look=10;
plot(time(1,1:look),G(1,1:look)),title('G(1,:)')
xlabel('(Sec)')
meta test6
plot(time(1,1:look),G(2,1:look)),title('G(2,:)')
xlabel('(Sec)')
meta test7
plot(time(1,1:look),G(3,1:look)),title('G(3,:)')
xlabel('(Sec)')
meta test8
plot(XKK(2,:)),title('Estimate of Angular rate')
xlabel('(Samples taken)')
meta test4
plot(XKK(3,:)),title('Estimate of angular acceleration')
xlabel('(Samples taken)')
meta test5
```

E. MISSILE2.M

This program was developed for tracking a target using fusion. The two sensors are located at the origin.

```
dt1=0.01;
dt2=1;
kmax=3000;

%Missile
speed=1;                    %approx. 3.3 time the speed of sound
xrange=0;                   %km
yrange=20;                  %km
x(:,1)=[xrange 3*speed/5 yrange -4*speed/5]';
theta(1,:)=atan2(x(1,1),x(3,1));
A=[0 1 0 0;0 0 0 0;0 0 0 1;0 0 0 0];
B=[0 0;1 0;0 0;0 1];
[phi,del]=c2d(A,B,dt1);

%Filter
i=1;
R=.01;
Q=0;
A=[0 1 0;0 0 1;0 0 0];
B=[0 0 1]';
H=[1 0 0];
[Phi,Del]=c2d(A,B,dt1);
XKKM1=zeros(3,1);
PKKM1=eye(3)*1e7;
Phi_IR=eye(3);
count=0;
for k=1:kmax
   G(:,k)=PKKM1*H'*(H*PKKM1*H'+R)^(-1);
   PKK=(eye(3)-G(:,k)*H)*PKKM1;
   PKKM1=Phi*PKK*Phi'+Q;
   XKK(:,k)=XKKM1(:,k)+G(:,k)*(theta(k)-XKKM1(1,k));
   if count*100+1 == k
      G(:,i)=PKKM1*H'*(H*PKKM1*H'+R)^(-1);
      PKK=(eye(3)-G(:,i)*H)*PKKM1;
      PKKM1=Phi_IR*PKK*Phi_IR'+Q;
      XKKM1(:,i)=XKK(:,k);
      XKK(:,i)=XKKM1(:,i)+G(:,i)*(theta(k)-XKKM1(1,i));
      XKKM1(:,k+1)=Phi*XKK(:,i);
      count=count+1;
      i=i+1;
   else
      XKKM1(:,k+1)=Phi*XKK(:,k);
   end
   x(:,k+1)=phi*x(:,k);
   theta(k+1)=atan2(x(1,k+1),x(3,k+1));
```

```
end
subplot(221)
plot(x(1,:),x(3,:)),grid,ylabel('(Y in (km))')
title('Missiles track'),xlabel('(X in (km))')
plot(theta*180/pi),grid,title('Bearing to the missile')
xlabel('Samples taken (t=.01)'),ylabel('Degrees')
plot(XKK(1,:)*180/pi),grid
title('Estimate (For Radar)')
xlabel('Samples taken (t=.01)'),ylabel('Degrees')
for k=1:500
   t(k)=dt1*k;
end
plot(t,G(1,1:500)),title('G(1,:)'),pause
plot(t,G(2,1:500)),title('G(2,:)')
plot(t,G(3,1:500)),title('G(3,:)')
plot(XKK(2,:)),title('Angular velocity')
plot(XKK(3,:)),title('Angular acceleration')
```

F. S_ELLIPSE.M

This program will produce error ellipsoids for the first few samples of the track.

```
echo off
!del pic3.met
load c:\matlab\wiseman\output2.dat
load c:\matlab\wiseman\truepos2.dat
load c:\matlab\wiseman\pkkout.dat
r=9;
axis('square')
axis([-50 500 -50 500])
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% plot path and estimate of path                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
plot(output2(1:r,1),output2(1:r,2),'+',
                    truepos2(1:r,2),truepos2(1:r,3))
xlabel('X - Position (ft)');ylabel('Y - Position (ft)')
title('Trajectory - S Curve')
gtext('_____  True track')
gtext('+ + +  Estimate')
gtext('O   O  Error Ellipsoids')
gtext('Enlarged 10 times')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% generate a uniformly spaced set of points                                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
delt=-1:.05:1;
delt=pi*delt;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% generate a (2xm) set of vectors on the unit circle                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
y=[cos(delt);sin(delt)];
shift=diag(ones(41));     % 41 points in delt
shift=shift';
hold on
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The function chol, to compute the Choleski                                    %
% decomposition, is done "backwards" in MATLAB. It                              %
% makes an upper triangular matrix R, not L. So take                            %
% the transpose.                                                                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for k=1:9
   i=k-1;
   P=pkkout(2*i+1:2*i+2,1:2);
   s=chol(P)';
   x=s*y;
   plot(x(1,:)*10+output2(k,1)*shift,
                          x(2,:)*10+output2(k,2)*shift)
end
```

```
%%%
```

G. TORPSCUR.M

This is the program to generate a true track and a monte carlo track. The program generates a simulation for a 40 knot torpedo following an s-shaped track.

Velocity is in yards/second and position is in feet. Data orientation is similar to actual data.

```
   ndim     = 3;
%
%
%%%%%  Establish Measurement Standard Deviations   %%%%%%%%
%
   sigmapos  = diag([ 15.0 , 15.0 , 0.00 ]) ;
   sigmavel  = diag([ 0.01 , 0.01 , 0.00 ]) ;
   sigmaacc  = diag([ 0.03 , 0.03 , 0.00 ]) ;
%
%%  Let User Input Number of "Noisy" Tracks to Generate   %%
%%      and the Seed for the Random Number Generator      %%
%
   nsimul     = input(['Enter the Base Reference Number for
                this Simulation (100 <= n <= 900): '])  ;
   nsimul    = fix(min(max(nsimul,100),900) ) ;
   fprintf(' Your Base Reference Number is - %4.0f\n',
            nsimul) ;
%
   ntrack      = input('Enter Number of Random Tracks to
                 Generate: ')  ;
   nseed = 0  ;
   if  ntrack > 0 ;
       nseed        = input('Enter Seed for Random Number
                      Generator: ');
   end
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%   The Following Section Generates the "True"      %%%%%
%%%%   and Simulated Tracks.                           %%%%%
%%%%   This Section Must Be Changed as Appropriate     %%%%%
%%%%   in Order to Generate Other Tracks               %%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%
%%%%%%%%%%%%%   Establish Model Constants   %%%%%%%%%%%%%%
%
   r        = 250;
   deltat   = 0.5;
   tfinal   = 45.0;
   kmax     = tfinal/deltat + 1;
   ydconv   = 3.0;
   v40kt    = 200/9 ;
   thetadot = ydconv*v40kt/r ;
%
```

```
%
%%%%%%%%%%%%    Initialize Data Arrays      %%%%%%%%%%%%%%%
%
   x      = zeros(kmax,ndim);
   v      = zeros(kmax,ndim);
   a      = zeros(kmax,ndim);
%
%%%%%     Generate Accelerations for a S-Track      %%%%
%
   v(1,1:2) = [ 30 30 ] ;
   theta  = thetadot*deltat*( (0:119 )'+0.5) ;
   a(25:34,1)    = -12.*ones(10,1);
   a(45:54,2)    = -12.*ones(10,1);
   a(65:74,1:2) =  ones(10,1)*[ 12. 12. ];
%
%
%%%%%   Then Generate Velocities and Positions by     %%%%%
%%%%%        Piecewise Constant Integration           %%%%%
%
%
   for i=2:kmax ;
       v(i,:)    = v(i-1,:) + a(i-1,:)*deltat/ydconv    ;
       x(i,:)    = x(i-1,:) + ydconv*v(i-1,:)*deltat ... ;
                      + a(i-1,:)*deltat^2/2   ;
   end
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%%%%%           Plot the "True" Track                  %%%%%
%
   eval( [ '! del data\track',num2str(nsimul),'.met' ])
   plot(x(1:kmax,1),x(1:kmax,2));xlabel('x-axis');
   ylabel('y-axis');grid;
   eval( [ 'meta data\track',num2str(nsimul) ])
%
%  and Save the True Track Position/Velocity/Acceleration %%
%
   eval( [ 'save data\delt',num2str(nsimul),'  deltat ' ])
   eval( [ 'save data\truep',num2str(nsimul),'  x  ' ])
   eval( [ 'save data\truev',num2str(nsimul),'  v  ' ])
   eval( [ 'save data\truea',num2str(nsimul),'  a  ' ])
%
%%%%%   Next Generate the Requested Number of "Noisy"  %%%%%
%%%%%   Tracks by Adding Random Noise with the         %%%%%
%%%%%   Previously Specified Standard Deviations to    %%%%%
%%%%%   the "True" Track. (This is done is a loop,     %%%%%
%%%%%   with each track graphed and saved as soon as   %%%%%
%%%%%   it is generated.)                              %%%%%
%
   truex = x ; truev = v ; truea = a ;
   rand('normal');
```

```
%
   if ntrack > 0 ;
     rand('seed',nseed);
     eval( [ 'save data\seed',num2str(nsimul),'  nseed ' ])
       for j=1:ntrack              ;     % Generate Tracks
       x = truex + rand(kmax,3)*sigmapos ;
       v = truev + rand(kmax,3)*sigmavel ;
       a = truea + rand(kmax,3)*sigmaacc ;
%                                        % Then Save them, using
%                                        % the 'eval' function
     eval( [ 'save data\tstp',num2str(nsimul+j),'  x  ' ])
     eval( [ 'save data\tstv',num2str(nsimul+j),'  v  ' ])
     eval( [ 'save data\tsta',num2str(nsimul+j),'  a  ' ])
%
     clg ;                               % Then Plot Them
     eval( [ '! del data\track',num2str(nsimul+j),'.met' ])

plot(truex(1:kmax,1),truex(1:kmax,2),x(1:kmax,1),x(1:kmax,2)
,'--'),title('x (-) and xrandom (--)') ;
xlabel('x-axis'),ylabel('y-axis'),grid
       eval( [ 'meta data\track',num2str(nsimul+j) ])
   end ; end
%
```

H. TRACKSIM.M

```
%
%    TRACKSIM.M
%        THIS PROGRAM GENERATES SIMULATED TORPEDO TRACKS FOR
%    TESTING THE KEYPORT KALMAN FILTERING. THIS PROGRAM AS
%    CURRENTLY WRITTEN GENERATES A SIMULATION FOR A 40 KNOT
%    TORPEDO ENTERING A 250 FT RADIUS CIRCLE AT A TURNING RATE
%    OF 15 DEGREES/SECOND.  THE VELOCITY IS IN YARDS/SECOND
%    AND POSITION IS IN FEET, AND THE DATA ORIENTATION IS
%    SIMILAR TO ACTUAL DATA. HOWEVER, THE PROGRAM IS DESIGNED
%    SO THAT ONE SIMPLY NEED CHANGE THE ACCELERATION DATA (IN
%    THE PORTION OF THE CODE WHERE INDICATED)
%
%
   ndim     = 3;
%
%
%%%%%%  Establish Measurement Standard Deviations   %%%%%%%%%
%
   sigmapos  = diag([ 15.0 , 15.0 , 0.00 ]) ;
   sigmavel  = diag([ 0.01 , 0.01 , 0.00 ]) ;
   sigmaacc  = diag([ 0.03 , 0.03 , 0.00 ]) ;
%
%%%  Let User Input Number of "Noisy" Tracks to Generate %%%
%%%      and the Seed for the Random Number Generator    %%%
%
   nsimul    = input(['Enter the Base Reference Number for
                      this Simulation (100 <= n <= 900): ']) ;
   nsimul    = fix(min(max(nsimul,100),900) ) ;
   fprintf('Your Base Reference Number is - %4.0f\n',nsimul);
%
   ntrack    = input('Enter Number of Random Tracks to
                       Generate: ')  ;
   nseed = 0  ;
   if  ntrack > 0 ;
       nseed     = input('Enter Seed for Random Number
                          Generator: ');
   end
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%       The Following Section Generates the "True"      %%%
%%%     and Simulated Tracks.                             %%%
%%%       This Section Must Be Changed as Appropriate     %%%
%%%     in Order to Generate Other Tracks                 %%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%%%%%%%%%%%%%%   Establish Model Constants   %%%%%%%%%%%%%%%
%
   r         = 250;
```

```
   deltat   = 0.5;
   tfinal   = 80.0;
   kmax     = tfinal/deltat + 1;
   ydconv   = 3.0;
   v40kt    = 200/9 ;
   thetadot = ydconv*v40kt/r ;
%%%%%%%%%%%%%%    Initialize Data Arrays     %%%%%%%%%%%%%%
%
   x      = zeros(kmax,ndim);
   v      = zeros(kmax,ndim);
   a      = zeros(kmax,ndim);
%
%
%%%%     Generate Accelerations for a Circular Track   %%%%%
%
   v(1,:) = [ v40kt , 0 , 0 ] ;
   theta  = thetadot*deltat*( (0:119 )'+0.5) ;
   a(20,1:2)     = [ -v40kt/deltat , v40kt/deltat ]
                    *ydconv  ;
   a(21:140,1:2) = ( (v40kt*ydconv)^2/r )
                    *[ -cos(theta)  , -sin(theta) ] ;
%
%
%%%%%   Then Generate Velocities and Positions by   %%%%%%%%%
%%%%%        Piecewise Constant Integration         %%%%%%%%%
%
%
   for i=2:kmax ;
       v(i,:)   = v(i-1,:) + a(i-1,:)*deltat/ydconv    ;
       x(i,:)   = x(i-1,:) + ydconv*v(i-1,:)*deltat ... ;
                    + a(i-1,:)*deltat^2/2   ;
   end                                      ;
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
%%%%%%           Plot the "True" Track           %%%%%%%%%%
%
   eval( [ '! del data\track',num2str(nsimul),'.met' ])
   plot(x(1:kmax,1),x(1:kmax,2));xlabel('x-axis');
   ylabel('y-axis');grid;
   eval( [ 'meta data\track',num2str(nsimul) ])
%
% and Save the True Track Position/Velocity/Acceleration  %%
%
   eval( [ 'save data\delt',num2str(nsimul),'  deltat ' ])
   eval( [ 'save data\truep',num2str(nsimul),'  x  ' ])
   eval( [ 'save data\truev',num2str(nsimul),'  v  ' ])
   eval( [ 'save data\truea',num2str(nsimul),'  a  ' ])
%
%%%%%   Next Generate the Requested Number of "Noisy"  %%%%%
%%%%%   Tracks by Adding Random Noise with the         %%%%%
```

```
%%%%%   Previously Specified Standard Deviations to    %%%%%
%%%%%   the "True" Track. (This is done is a loop,     %%%%%
%%%%%   with each track graphed and saved as soon as   %%%%%
%%%%%   it is generated.)                              %%%%%
%
   truex = x ; truev = v ; truea = a ;
   rand('normal');
%
   if ntrack > 0 ;
       rand('seed',nseed);
       eval( [ 'save data\seed',num2str(nsimul),'  nseed ' ])
       for j=1:ntrack              ;     % Generate Tracks
       x = truex + rand(kmax,3)*sigmapos ;
       v = truev + rand(kmax,3)*sigmavel ;
       a = truea + rand(kmax,3)*sigmaacc ;
%                                                  % Then Save
them, using
%                                                 % the 'eval'
function
       eval( [ 'save data\tstp',num2str(nsimul+j),'  x  ' ])
       eval( [ 'save data\tstv',num2str(nsimul+j),'  v  ' ])
       eval( [ 'save data\tsta',num2str(nsimul+j),'  a  ' ])
%
       clg ;                                  % Then Plot Them
%
       eval( [ '! del data\track',num2str(nsimul+j),'.met' ])

plot(truex(1:kmax,1),truex(1:kmax,2),x(1:kmax,1),x(1:kmax,2)
    ,'--') ;
title('x (-) and xrandom (--)') ;
xlabel('x-axis'),ylabel('y-axis'),grid

       eval( [ 'meta data\track',num2str(nsimul+j) ])
   end ; end
%
```

APPENDIX C: THE KALMAN FILTER AND MANEUVER DETECTION

A. OBSDATA.FOR

```
C   OBSDATA.FOR
C   MODIFICATIONS MADE BY ART SCHOENSTADT ON AUGUST 14, 1992.
C
C---THIS PROGRAM IS DESIGNED TO PROCESS 3/D DATA FILES WHICH
C---SIMULATE THE UNDERSEA TRACKING RANGE AT KEYPORT, WA.
C---THE INPUT DATA FILES ARE GENERATED BY A MATLAB SIMULATION.
C---THIS PROGRAM IS DESIGNED TO RUN ON THE IBM/AT PERSONAL
C---COMPUTER. THE USER IS ALSO REQUIRED TO PROVIDE THE INPUT
C---DATA FILES IN SPECIFIC FORMAT, AND THE NUMBER OF POINTS TO
C---CALCULATE (MAXIMUM OF 2500).  THE PROGRAM USES A
C---LARGE AMOUNT OF HARD DISK SPACE FOR TEMPORARY
C---FILES (APPROXIMATELY 160K BYTES/100 POINTS, 4MEG BYTES MAX
C---FOR 2500 POINTS.
C---THE PROGRAM ALSO INCLUDES "HARD WIRED" CONVERSIONS TO
C---REFLECT THE FACT THAT SOME DATA IS IN FEET, OTHER DATA IN
C---YARDS, AND DEPTH IS MEASURED IN POSITIVE, NOT NEGATIVE
C---TERMS.
C
C****************     SECTION 1    ***************************
C
C*** DECLARATION OF PARAMETERS ***
C*** MAXOBS IS THE MAXIMUM NUMBER OF OBSERVATIONS THAT    ***
C***             CAN BE USED                              ***
C*** MAXSTATE IS SET FOR A 3-DIMENSIONAL, POSITION,       ***
C***            VELOCITY AND ACCELERATION PLANT           ***
C*** DEPS IS USED TO TEST WHEN TWO OBSERVATIONS ARE CLOSE ***
C***            ENOUGH TO BE CONSIDERED SIMULTANEOUS      ***
C
      PARAMETER( MAXOBS=250, MAXSTATE=9 )
      PARAMETER( DEPS=0.005D0, STEP=0.5D0 )
C
C     ***** DECLARATION OF VARIABLES *****
C
      DOUBLE PRECISION PTC(MAXOBS),VTIME(MAXOBS),
     *    SOURCE(MAXOBS),ZK(MAXSTATE,MAXOBS),ROLL(MAXOBS)
      DOUBLE PRECISION SS,SSV,SEC,X,Y,Z,XVEL,
     *   YVEL,ZVEL,ACCX,ACCY,ACCZ,EYAW,EROLL,EPITCH,PTM,VTM,
     *    ATM,RTM,ATM1,ACCXM1,ACCYM1,ACCZM1,RLLM1,SSA,SSR,
     *    TORANGE,TOTORP,TNEXTOBS
C
      INTEGER HH,HHV,MM,MMV,HOUR,MIN,PC,MMA,MMR,NOBSERV,SRC
C
```

```
       CHARACTER ASK*1
C      CHARACTER PKKS3D*13, PKKM1S*13, INFILC*13, INFILR*13
C      CHARACTER INFILP*13, INFILA*13, OUTFIL*13
C
C
       LOGICAL*1  POSREC,VELREC,ACCREC,ENDPOSDAT,ENDVELDAT,
     *            ENDACCDAT,ENDRLLDAT
C
C      ****  INPUT DATA, DESIGNATE FILENAMES  ****
C
       OPEN(1, FILE= 'DATA\POSITION.OBS', STATUS= 'OLD')
       OPEN(2, FILE= 'DATA\VELOCITY.OBS', STATUS= 'OLD')
       OPEN(3, FILE= 'DATA\ACCELERA.OBS', STATUS= 'OLD')
       OPEN(4, FILE= 'DATA\ROLL_YAW.OBS', STATUS= 'OLD')
       OPEN(11,FILE= 'DATA\OBSERVAT.DAT', STATUS= 'NEW')
C
C
C      **** NOTIFY USER OF MAXIMUM NUMBER OF DATA POINTS ****
C
       WRITE(*,770)  MAXOBS
  770  FORMAT(' The MAXIMUM number of points that can be ',
     *      'analyzed is:',I4)
       WRITE(*,771)
  771  FORMAT(' To use more, you must reset the parameter ',
     *      'MAXOBS',/,' in the main program and recompile')
C
C
C
C***************     SECTION 2    ***************************
C
C      **** GET STARTING TIME FOR RANGE DATA ****
C
       READ(1,750,END=1200) PC,X,Y,Z,HH,MM,SS
       WRITE(*,751) HH,MM,SS
       TORANGE = 3600.*REAL(HH)+60.*REAL(MM)+SS
       WRITE(*,752)
       READ(*,755) HOUR,ASK,MIN,SEC
       IF (ASK .EQ. ' ') GOTO 754
          TORANGE = 3600.*REAL(HOUR)+60.*REAL(MIN)+SEC
          WRITE(*,756) HOUR,MIN,SEC
  754  CONTINUE
C
  750  FORMAT(1X,I6,2X,3F10.1,10X,I2,1X,I2,1X,F5.2)
  751  FORMAT(/,' The first position time
     *is:',I2,':',I2,':',F7.4)
  752  FORMAT(/,' If you do not wish to change it, press
     *"Enter"',/,' If you wish to change it, enter the new
     *time in the format:  ',/,' HH:MM:SS.SSSS')
  755  FORMAT(I2,A1,I2,1X,F7.4)
  756  FORMAT(' The first position time is
     *now:',I2,':',I2,':',F7.4)
```

```
C
C     **** GET STARTING TIME FOR TORPEDO DATA ****
C
      READ(2,760,END=1201) MMV,SSV,XVEL,YVEL,ZVEL
      HHV = 0
      T0TORP = 60.*REAL(MMV)+SSV
      WRITE(*,761) HHV,MMV,SSV
      WRITE(*,752)
      READ(*,755,END=763) HOUR,ASK,MIN,SEC
      IF (ASK .EQ. ' ')  GOTO 763
         T0TORP = 3600.*REAL(HOUR)+60.*REAL(MIN)+SEC
         WRITE(*,762) HOUR,MIN,SEC
  763 CONTINUE
C
  760 FORMAT(1X,I2,1X,F7.4,10X,F9.3,5X,F9.3,5X,F9.3)
  761 FORMAT(/,' The first velocity time
     *is:',I2,':',I2,':',F7.4)
  762 FORMAT(' The first velocity time is
     *now:',I2,':',I2,':',F7.4)
C
C
C ****   SELECT THE FIRST RECORD FROM EACH DATA SET THAT  ****
C **** REPRESENTS AN OBSERVATION AT OR AFTER THE STARTING ****
C **** TIME. NOTE THAT WE MUST USE A SLIGHTLY DIFFERENT   ****
C **** PROCEDURE FOR THE POSITION AND VELOCITY DATA       ****
C **** FILES, SINCE WE HAVE ALREADY READ A RECORD FROM    ****
C **** EACH OF THEM.                                      ****
C
  201 CONTINUE
      IF ( PTM .GE. T0RANGE ) GOTO 202
      READ(1,750,END=1200) PC,X,Y,Z,HH,MM,SS
         PTM = 3600.*REAL(HH)+60.*REAL(MM)+SS
         GOTO 201
C
  202 CONTINUE
      IF ( VTM .GE. T0TORP ) GOTO 203
      READ(2,760,END=1201) MMV,SSV,XVEL,YVEL,ZVEL
         VTM = 60.*REAL(MMV)+SSV
         GOTO 202
C
  203 CONTINUE
      READ(3,760,END=1202) MMA,SSA,ACCX,ACCY,ACCZ
      ATM = 60.*REAL(MMA)+SSA
      IF ( ATM .LT. T0TORP ) GOTO 203
         ATM1=ATM
         ACCXM1=ACCX
         ACCYM1=ACCY
         ACCZM1=ACCZ
C
  204 CONTINUE
      READ(4,760,END=1203) MMR,SSR,EYAW,EROLL,EPITCH
```

```
      RTM = 60.*REAL(MMR)+SSR
      IF ( RTM .LT. T0TORP ) GOTO 204
         RTM1=RTM
         RLLM1=EROLL
C
C
C   ****        SYNCHRONIZE RANGE AND TORPEDO CLOCKS  ******
C
      PTM     = PTM - T0RANGE
      VTM     = VTM - T0TORP
      ATM     = ATM - T0TORP
      RTM     = RTM - T0TORP
C
C****************     SECTION 3    *************************
C
C***    BUILD THE OBSERVATION ARRAY BY SEQUENTIALLY        ***
C***  DETERMINING THE NEXT OBSERVATION TIME AND THE        ***
C***    SPECIFIC FIELDS OBSERVED AT THAT TIME              ***
C
C
      WRITE(*,*)  ' '
      WRITE(*,*)  'Reading Data Files'
      NOBSERV  = 0
      ENDPOSDAT = .FALSE.
      ENDVELDAT = .FALSE.
      ENDACCDAT = .FALSE.
      ENDRLLDAT = .FALSE.
C
  300 CONTINUE
      IF (ENDVELDAT .AND. ENDPOSDAT )    GOTO 400
      IF (NOBSERV.GE.MAXOBS)   GOTO 395
C
      NOBSERV = NOBSERV + 1
      POSREC = .FALSE.
      VELREC = .FALSE.
      ACCREC = .FALSE.
      TNEXTOBS = DMIN1(PTM,VTM)
      IF (ENDPOSDAT)     TNEXTOBS = VTM
      IF (ENDVELDAT)     TNEXTOBS = PTM
      VTIME(NOBSERV) = TNEXTOBS
C
C***   SEE IF THERE'S A POSITION OBSERVATION AT THIS TIME ***
C
  311 CONTINUE
      IF ( ( (PTM-TNEXTOBS) .LT. DEPS ) .AND.
     *        ( .NOT.(ENDPOSDAT) ) ) THEN
         POSREC = .TRUE.
         PTC(NOBSERV)=PC
         ZK(1,NOBSERV)=X
         ZK(4,NOBSERV)=Y
         ZK(7,NOBSERV)=Z
```

```
         READ(1,750,END=312) PC,X,Y,Z,HH,MM,SS
         PTM = 3600.*REAL(HH)+60.*REAL(MM)+SS - T0RANGE
         IF ( PTM .LT. TNEXTOBS ) GOTO 1210
       ENDIF
       GOTO 321
C
  312 ENDPOSDAT=.TRUE.
       WRITE(*,313) NOBSERV
  313 FORMAT(' End of Position Record File Encountered at ',
     *       'Observation ',I4)
C
C
C***   SEE IF THERE'S A VELOCITY OBSERVATION AT THIS TIME ***
C
  321 CONTINUE
       IF ( ( (VTM-TNEXTOBS) .LT. DEPS ) .AND.
     *         ( .NOT.(ENDVELDAT) ) )   THEN
         VELREC = .TRUE.
         ZK(2,NOBSERV)= 3.*XVEL
         ZK(5,NOBSERV)=-3.*YVEL
         ZK(8,NOBSERV)=-3.*ZVEL
         READ(2,760,END=322) MMV,SSV,XVEL,YVEL,ZVEL
         VTM = 60.*REAL(MMV)+SSV - T0TORP
         IF ( VTM .LT. TNEXTOBS ) GOTO 1211
       ENDIF
       GOTO 331
C
  322 ENDVELDAT=.TRUE.
       WRITE(*,323) NOBSERV
  323 FORMAT(' End of Velocity Record File Encountered at ',
     *       'Observation ',I4)
C
C
C* SEE IF THERE'S AN ACCELERATION OBSERVATION AT THIS TIME  *
C* THEN USE THE CLOSEST ACCELERATION OBSERVATION TO THE     *
C* CURRENT TIME.                                            *
C
  331 CONTINUE
       IF ( ( ATM .LE. TNEXTOBS ) .AND.
     *         ( .NOT.(ENDACCDAT) ) ) THEN
         ATM1=ATM
         ACCXM1=ACCX
         ACCYM1=ACCY
         ACCZM1=ACCZ
  332    READ(3,760,END=333) MMA,SSA,ACCX,ACCY,ACCZ
         ATM = 60.*REAL(MMA)+SSA - T0TORP
         IF ( ATM .LE. TNEXTOBS ) GOTO 332
       ENDIF
       GOTO 335
C
  333 ENDACCDAT=.TRUE.
```

```
      WRITE(*,334) NOBSERV
  334 FORMAT(' End of Acceleration Record File Encountered at
     *Observation ',I4)
C
  335 CONTINUE
      IF ( (TNEXTOBS-ATM1) .LT. (ATM - TNEXTOBS) ) THEN
            ZK(3,NOBSERV)=ACCXM1
            ZK(6,NOBSERV)=ACCYM1
            ZK(9,NOBSERV)=ACCZM1
         ELSE
            ZK(3,NOBSERV)=ACCX
            ZK(6,NOBSERV)=ACCY
            ZK(9,NOBSERV)=ACCZ
      ENDIF
C
C***  SEE IF THERE'S A ROLL OBSERVATION AT THIS TIME    *****
C***    THEN USE THE CLOSEST ROLL OBSERVATION TO THE    *****
C***    CURRENT TIME.                                   *****
C
C
  341 CONTINUE
      IF ( ( RTM .LE. TNEXTOBS ) .AND.
     *         ( .NOT.(ENDRLLDAT) ) ) THEN
         RTM1=RTM
         RLLM1=EROLL
  342    READ(4,760,END=343) MMR,SSR,EYAW,EROLL,EPITCH
         RTM = 60.*REAL(MMR)+SSR - T0TORP
         IF ( RTM .LT. TNEXTOBS ) GOTO 342
      ENDIF
      GOTO 345
  343 ENDRLLDAT=.TRUE.
      WRITE(*,344) NOBSERV
  344 FORMAT(' End of Roll Record File Encountered at ',
     *       'Observation ',I4)
C
  345 CONTINUE
      IF ( (TNEXTOBS-RTM1) .LT. (RTM - TNEXTOBS) ) THEN
            ROLL(NOBSERV)=RLLM1
         ELSE
            ROLL(NOBSERV)=EROLL
      ENDIF
      ACCREC = .TRUE.
C
C****   CODE THE SOURCE(S) OF THIS RECORD          ******
C
  351 CONTINUE
      SRC = 0
      IF ( POSREC )    SRC = 73
      IF ( VELREC )    SRC = SRC + 146
C      IF ( ACCREC )    SRC = SRC + 292
      SOURCE(NOBSERV)= SRC
```

```
C
C
      GOTO 300
C
  395 WRITE(*,396)  NOBSERV,MAXOBS
  396 FORMAT(' The total number of observations (',I4,')
     *exceeds allocate storage (MAXOBS = ',I4,/,5x,'Some data
     *will be lost. You may wish to recompile the program',/,
     * with a higher value of MAXOBS. (Enter "Y" to
     *continue)')
      READ(*,397)  ASK
  397 FORMAT(A1)
      IF ( ( ASK .NE. 'Y' ) .AND. ( ASK .NE. 'y' ) )  STOP
C
C
C*****************    SECTION 4    ***************************
C****            WRITE OUT DATA FILES                        ****
C
C
  400 CONTINUE
      CLOSE(1)
      CLOSE(2)
      CLOSE(3)
      CLOSE(4)
      WRITE(*,902)  NOBSERV
      WRITE(11,901) NOBSERV
      DO  401  IPRT = 1,NOBSERV
          WRITE(11,900) PTC(IPRT),VTIME(IPRT),SOURCE(IPRT),
     *      ( ZK(JPRT,IPRT),JPRT=1,MAXSTATE),ROLL(IPRT)
  401 CONTINUE
  900 FORMAT(13(1X,E14.8))
  902 FORMAT(/,' Writing ',I5,' Combined Observations to
     *File')
  901 FORMAT('  OUTPUT FROM PROGRAM OBSDATA.FOR ',/,
     *1X,I4,' OBSERVATIONS',
     */,7X,'PTC',11X,'TIME',10X,'SOURCE',10X,'ZK(1)',10X,
     *'ZK(2)',10X,'ZK(3)',10X,'ZK(4)',10X,'ZK(5)',10X,'ZK(6)',
     *10X,'ZK(7)',10X,'ZK(8)',10X,'ZK(9)',10X,'ROLL')
      CLOSE(11)
C
C
      STOP
C
C  ****   ERROR ROUTINES FOR UNEXPECTED END OF DATA FILE  ****
C
 1200 WRITE(*,1300)
 1300 FORMAT('**** ERROR - RANGE DATA FILE ENDED BEFORE A
     *VALID ','RECORD COULD BE LOCATED')
      STOP
C
 1201 WRITE(*,1301)
```

```
 1301 FORMAT('**** ERROR - VELOCITY DATA FILE ENDED BEFORE A
     *VALID RECORD COULD BE LOCATED')
      STOP
C
 1202 WRITE(*,1302)
 1302 FORMAT('**** ERROR - ACCELERATION DATA FILE ENDED BEFORE
     *A VALID RECORD COULD BE LOCATED')
      STOP
C
 1203 WRITE(*,1303)
 1303 FORMAT('**** ERROR - ROLL DATA FILE ENDED BEFORE A VALID
     *RECORD COULD BE LOCATED')
      STOP
C
C
 1210 WRITE(*,1310)
 1310 FORMAT('**** ERROR - RANGE DATA FILE RECORD OUT OF VALID
     *TIME ORDER')
      STOP
C
 1211 WRITE(*,1311)
 1311 FORMAT('**** ERROR - VELOCITY DATA FILE OUT OF VALID
     *TIME ORDER')
      STOP
C
 1212 WRITE(*,1312)
 1312 FORMAT('**** ERROR - ACCELERATION DATA FILE OUT OF VALID
     *TIME ORDER')
      STOP
C
 1213 WRITE(*,1313)
 1313 FORMAT('**** ERROR - ROLL DATA FILE OUT OF VALID
     *TIME ORDER')
      STOP
C
      END
```

B. POSCONV.FOR

```
C
C     POSCONV.FOR
C
C     THIS PROGRAM CONVERTS THE DATA FILES CREATED BY THE
C     MATLAB SIMULATION FOR POSITION, VELOCITY, AND
C    ACCELERATION INTO THE PROPER FORMAT FOR USE BY THE KALMAN
C     FILTER PROGRAM.  IT ALSO CREATES A ROLL ANGLE DATA FILE
C     OF 0 DEGREES THROUGHOUT.
C
       INTEGER PC,MIN,HR
       DOUBLE PRECISION  SEC,X,Y,Z,VX,VY,VZ,AX,AY,AZ,YAW,
       DOUBLE TRECISION  ROLL,PITCH,TOTSEC,DELTAT
       LOGICAL*1   ENDPOSREC, ENDVELREC, ENDACCREC
C
C *******   OPEN THE TIMING INFORMATION DATA FILE  ***********
      OPEN(1,FILE='DATA\DELTSIMU',STATUS='OLD')
C
C *******   OPEN THE INPUT POSITION DATA FILE      ***********
      OPEN(2,FILE='DATA\SIMULPOS',STATUS='OLD')
C
C ******   OPEN THE INPUT VELOCITY DATA FILE       ***********
      OPEN(3,FILE='DATA\SIMULVEL',STATUS='OLD')
C
C ******   OPEN THE INPUT ACCELERATION DATA FILE   ***********
      OPEN(4,FILE='DATA\SIMULACC',STATUS='OLD')
C
C
C
C ******   OPEN THE OUTPUT POSITION DATA FILE      ***********
      OPEN(12,FILE='DATA\POSITION.OBS',STATUS='NEW')
C
C ******   OPEN THE OUTPUT VELOCITY DATA FILE      ***********
      OPEN(13,FILE='DATA\VELOCITY.OBS',STATUS='NEW')
C
C ******   OPEN THE OUTPUT ACCELERATION DATA FILE  ***********
      OPEN(14,FILE='DATA\ACCELERA.OBS',STATUS='NEW')
C
C ******   OPEN THE OUTPUT ROLL ANGLE DATA FILE    ***********
      OPEN(15,FILE='DATA\ROLL_YAW.OBS',STATUS='NEW')
C
C
C  ************************************************************
C  ***     INPUT THE MATLAB FILES AND CONVERT TO INPUT     ****
C  ***          FORMAT FOR THE FILTER PROGRAM              ****
C  ************************************************************
C
       PC=1
```

```
      TOTSEC= 0.D0
      READ(1,20)  DELTAT
C
C
      ENDPOSREC = .FALSE.
      ENDVELREC = .FALSE.
      ENDACCREC = .FALSE.
  100 CONTINUE
      IF  ( ENDPOSREC .AND. ENDVELREC .AND. ENDACCREC )
     *           GOTO  200
C
C  ***          UPDATE TIMING DATA                          ***
C
          SEC = AMOD(TOTSEC,60.0)
          MIN = MOD( DINT(TOTSEC/60.D0), 60 )
          HR  = DINT(TOTSEC/3600.D0)
C
C  ***          PROCESS ROLL, YAW AND PITCH DATA            ***
C
          YAW=.00
          ROLL=.00
          PITCH=.00
          WRITE(15,40) MIN,SEC,YAW,ROLL,PITCH
C
C  ***          PROCESS POSITION DATA                       ***
C
          IF  ( .NOT. ENDPOSREC )  THEN
              READ(2,20,END=101) X,Y,Z
              WRITE(12,30) PC,X,Y,Z,HR,MIN,SEC
          ENDIF
C
C  ***          PROCESS VELOCITY DATA                       ***
C
          IF  ( .NOT. ENDVELREC )  THEN
              READ(3,20,END=102) VX,VY,VZ
              VY=-VY
              VZ=-VZ
              WRITE(13,40) MIN,SEC,VX,VY,VZ
          ENDIF
C
C  ***          PROCESS ACCELERATION DATA                   ***
C
          IF  ( .NOT. ENDACCREC )  THEN
              READ(4,20,END=103) AX,AY,AZ
              WRITE(14,40) MIN,SEC,AX,AY,AZ
          ENDIF
C
C  ***          UPDATE COUNTERS AND TIMING                  ***
C
          PC=PC+1
          TOTSEC=TOTSEC+DELTAT
```

```
C
          GOTO  100
C
C
C
   20     FORMAT(3(1X,E15.7))
   30     FORMAT(1X,I6,2X,3F10.1,10X,I2,1X,I2,1X,F5.2)
   40     FORMAT(1X,I2,1X,F7.4,10X,F9.3,5X,F9.3,5X,F9.3)
C
  101 CONTINUE
      WRITE(*,121) PC-1
      ENDPOSREC = .TRUE.
      GOTO 100
  121 FORMAT( ' End of Data After ',I4,'  Position Records
     *Read')
C
  102 CONTINUE
      WRITE(*,122) PC-1
      ENDVELREC = .TRUE.
      GOTO 100
  122 FORMAT( ' End of Data After ',I4,'  Velocity Records
     *Read')
C
  103 CONTINUE
      WRITE(*,123) PC-1
      ENDACCREC = .TRUE.
      GOTO 100
  123 FORMAT( ' End of Data After ',I4,'  Acceleration Records
     *Read')
C
  200 CONTINUE
      CLOSE(1)
      CLOSE(2)
      CLOSE(3)
      CLOSE(4)
      CLOSE(12)
      CLOSE(13)
      CLOSE(14)
      CLOSE(15)
      STOP
      END
```

C. TORPMAN.FOR

```
C--KALMAN FILTER---KALMAN SMOOTHING ALGORITHM-------------
C--THIS PROGRAM IS DESIGNED TO PROCESS 3/D DATA FILES FROM
C--THE UNDERSEA TRACKING RANGE AT KEYPORT WA. A KALMAN
C--FILTER IS APPLIED TO THE TRACK DATA WHICH CONSISTS OF
C--X,Y, AND Z COORDINATES, AS WELL AS VELOCITY COMPONENT IN
C--X,Y, AND Z COORDINATES. THEN, A KALMAN FILTER SMOOTHING
C--ROUTINE GENERATES SMOOTHED POINTS IN X,Y, AND Z. THE
C--PROGRAM GENERATES OUTPUT FILES WHICH CONTAIN THE VARIANCE
C--OF THE X ESTIMATE VS SAMPLE FOR BOTH THE FORWARD KALMAN
C--FILTER CASE AND THE KALMAN SMOOTHED CASE. FILES ARE ALSO
C--GENERATED WHICH CONTAIN THE FILTERED X,Y,AND Z ESTIMATES
C--AND THE SMOOTHED X,Y, AND Z ESTIMATES. THIS PROGRAM IS
C--DESIGNED TO RUN ON THE IBM/AT PERSONAL COMPUTER BUT DUE
C--TO THE SIZE OF THE DATA SETS INVOLVED, PLOTTING CANNOT BE
C--DONE WITH THIS PROGRAM. PLOTTING OF OUTPUT DATA IS
C--DONE USING MATLAB. THE PROGRAM GIVES THE USER THE OPTION
C--OF CHANGING THE STARTING TIMES FOR BOTH THE RANGE AND
C--VELOCITY DATA, THE VALUE OF THE INITIAL COVARIANCE
C--MATRIX, AND THE EXCITATION PROCESS VECTOR. THE USER IS
C--ALSO REQUIRED TO PROVIDE THE NAMES OF THE INPUT AND
C--OUTPUT DATA FILES, AND THE NUMBER OF POINTS TO CALCULATE
C--WITH A MAXIMUM OF 2500.  THE PROGRAM USES A LARGE AMOUNT
C--OF HARD DISK SPACE FOR TEMPORARY FILES (APPROXIMATELY
C--160K BYTES/100 POINTS, 4MEG BYTES MAX FOR 2500 POINTS).
C
C************************** SECTION 1 **********************
C
C ***************    DECLARATION OF PARAMETERS    ***********
C
      INTEGER   ONE
      PARAMETER ( ONE=1, MAXOBS=250 , NDIMEN=3 )
      PARAMETER ( NUMSTA=3*NDIMEN, NSTMAT=NUMSTA*NUMSTA,
     *            NOBVEC=NUMSTA*MAXOBS )
C
C ***************       DECLARATION OF ARRAYS     ************
C
      DOUBLE PRECISION    XKK(NUMSTA),XKKM1(NUMSTA),
     *    ZZ(NUMSTA)ZKKM1(NUMSTA),XNNM1(NUMSTA),XP1(NUMSTA)
      DOUBLE PRECISION    TMPV1(NUMSTA),TMPV2(NUMSTA)
      DOUBLE PRECISION    VTIME(MAXOBS),SOURCE(MAXOBS),
     *    ROLL(MAXOBS)
      DOUBLE PRECISION    XKKS(NUMSTA,MAXOBS)

C
```

```
      DOUBLE PRECISION    PHI(NUMSTA,NUMSTA),
     *     Q(NUMSTA,NUMSTA),HI(NUMSTA,NUMSTA),
     *     PKKM1(NUMSTA,NUMSTA),PKK(NUMSTA,NUMSTA),
     *     H(NUMSTA,NUMSTA),HTRANS(NUMSTA,NUMSTA),
     *     R(NUMSTA,NUMSTA),G(NUMSTA,NUMSTA),
     *     PHIT(NUMSTA,NUMSTA),PKKS(NUMSTA,NUMSTA),
     *     PKKP1(NUMSTA,NUMSTA),AK(NUMSTA,NUMSTA),
     *     RPROJ(NUMSTA,NUMSTA)
      DOUBLE PRECISION    TEMP1(NUMSTA,NUMSTA),
     *     TEMP2(NUMSTA,NUMSTA),TEMP3(NUMSTA,NUMSTA)
C
C ***********     DECLARATION OF SCALARS     ***************
C
      DOUBLE PRECISION  ACCTHRSH,ROLTHRSH,WTMIN,WTMAX
      COMMON /CBLK/ PTC(MAXOBS),XYRANGE(3,2),
     *     PKKONEONE(MAXOBS),IR1
C
      INTEGER KF,SRC
C
      CHARACTER PKKS3D*13, PKKM1S*13
C     CHARACTER ASK*1, INFILC*13, INFILR*13, INFILP*13,
C    *     INFILA*13, OUTFIL*13
C
C
C *******    INITIALIZATION OF SELECTED ARRAYS  ************
C ******     (THOSE NOT OTHERWISE INITIALIZED) *************
C
      DATA  HI /NSTMAT*0.D0/, VTIME /MAXOBS*0.D0/,
     *  XKKS /NOBVEC*0.D0/,  SOURCE /MAXOBS*0.D0/,
     *  PKKM1 /NSTMAT*0.D0/, ROLL /MAXOBS*0.D0/
C
C ********************** SECTION 2 *************************
C
C *****  DESIGNATE AND OPEN INPUT AND OUTPUT FILES  ********
C
C     WRITE(*,*) 'ENTER NAME OF INPUT RANGE POSITION DATA
C    *FILE'
C     READ(50,'(A)') INFILP
C     WRITE(*,*) 'ENTER NAME OF INPUT INTERNAL VELOCITY DATA
C    *FILE'
C     READ(51,'(A)') INFILA
C     WRITE(*,*) 'ENTER NAME OF INPUT INTERNAL ACCELERATION
C    *DATA FILE'
C     READ(52,'(A)') INFILC
C     WRITE(*,*) 'ENTER NAME OF INPUT ROLL ANGLE DATA FILE'
C     READ(53,'(A)') INFILR
C     WRITE(*,*) 'ENTER NAME OF SMOOTHED DATA FILE'
C     READ(54,'(A)') OUTFIL
      OPEN(2,FILE= 'DATA\XKKFWD.DAT',  STATUS='NEW')
      OPEN(3,FILE= 'DATA\XKSMOOTH.DAT',STATUS='NEW')
      OPEN(4,FILE= 'DATA\PKKOUT.DAT',  STATUS='NEW')
```

```
      OPEN(11,FILE='DATA\OBSERVAT.DAT',STATUS='OLD')
      PKKS3D = 'DATA\$3DTEMP'
      PKKM1S = 'DATA\$M1TEMP'
      OPEN(63,FILE=PKKS3D, ACCESS='DIRECT',
     *     STATUS= 'NEW',FORM='FORMATTED',RECL=135)
      OPEN(61,FILE=PKKM1S, ACCESS='DIRECT',
     *     STATUS= 'NEW',FORM='FORMATTED',RECL=135)
C
C  *********   INITIALIZE SELECTED COUNTERS   *************
C
      IR1 = 1
      IR  = 1
      K   = 0
C
C  ****   INITIALIZE ARRAY DIMENSIONS AND VARIABLES  ******
C
      L  = 1
      N  = 9
      NW = 3
      ND = NUMSTA
      LD = ONE
C
      PKKM1(1,1) = 1000000.0D0
      ROLTHRSH=5.0
      ACCTHRSH=5.0
      WTMIN=1800.
      WTMAX = 20000.0
C
C *****   INITIALIZE THE IDENTITY MATRIX AND  X(1|0) ******
C
      DO 190 I = 1, 9
         XKKM1(I) = 0.0
         HI(I,I) = 1.
  190 CONTINUE
C
C *****   ECHO INITIAL FILTER PARAMETERS AND ALLOW   *****
C *****     USER TO CHANGE THEM IF DESIRED.          *****
C
      CALL CHANGE(PKKM1,WTMIN,WTMAX,N)
      CALL SETTHRSH(ROLTHRSH,ACCTHRSH)
C
C
C ********************** SECTION 3 **************************
C
C ****  READ IN THE "RAW" OBSERVATION DATA FILE, WHICH  ***
C ****  HAS BEEN PROPERLY FORMATTED IN AN EARLIER        ***
C ****  E.G. OBSDATA.FOR.  WHEN THIS DATA IS READ IN     ***
C ****  THE Kth OBSERVATION IS ORIGINALLY LOADED INTO    ***
C ****  XKKS(*,K). THEN, WHEN THE FILTER IS RUN, THIS    ***
C ****  VALUE WILL BE REPLACED WITH THE ESTIMATED STATE ***
C ****  X(K|K). FINALLY, WHEN THE BACKWARD SMOOTHING IS ***
```

```
C ****  RUN, THIS VALUE WILL BE REPLACED WITH THE FINAL ***
C *****  SMOOTHED VALUE.                                  ***
C
      READ(11,210)   IR1
      IF  ( IR1 .GT. MAXOBS )  THEN
          WRITE(*,209)  IR1, MAXOBS
          STOP
      ENDIF
C
      DO  201  IPRT = 1,IR1
          READ(11,211,END=300) PTC(IPRT),VTIME(IPRT),
     *      SOURCE(IPRT),( XKKS(JPRT,IPRT),JPRT=1,9),
     *      ROLL(IPRT)
  201 CONTINUE
      CLOSE(11)
C
  209 FORMAT(' The Number of Observations ',I5,', exceeds
     *the',' maximum storage available (',I5,')' )
  210 FORMAT(/,1X,I4,/)
  211 FORMAT(13(1X,E14.8))
  300 CONTINUE
C
C
C******************** SECTION 4 ***************************
C
C ************************************************************
C ****    IMPLEMENT THE (FORWARD) KALMAN FILTER      ******
C ************************************************************
C
  100 K = K + 1
      WRITE(*,*)  'Forward Filter - Step',K
C
C     ***  BYPASS COMPUTATION OF P(1|0) AND X(1|0)      ***
C     ***  DURING FIRST ITERATION                       ***
C
      IF  ( K .EQ. 1 )  GO TO 101
C
C     ***  GENERATE THE APPROPRIATE STATE TRANSITION    ***
C     ***  MATRIX  ( PHI ) AND ITS TRANSPOSE, BASED ON  ***
C     ***  THE ELAPSED TIME SINCE THE  LAST OBSERVATION.***
C
          DT = VTIME(K)-VTIME(K-1)
          CALL PHIDEL(DT,NW,PHI,ND)
          CALL TRANS(PHI,N,N,PHIT,ND,ND)
C
C     ****  DETERMINE WHETHER THE FILTER BELIEVES THE   ****
C     ****  TORPEDO IS MANEUVERING, AND GENERATE THE    ****
C     ****  APPROPRIATE COVARIANCE OF EXCITATION MATRIX ****
C
          CALL MANDET(ROLL,XKKS,K,ROLTHRSH,ACCTHRSH,WTMIN,
     *          WTMAX,DT,Q,ND,NW)
```

```
C
C     ***  FORM P(K|K-1), BASED ON THE EQUATION          ***
C
C           P(K|K-1) = PHI*P(K-1|K-1)*TRANS(PHI) + Q(K)
C
C     ***   WHERE  P(K-1|K-1)  IS P(K|K)  FROM THE       ***
C     ***    PREVIOUS STEP.                              ***
C
          CALL PROD(PKK,PHIT,N,N,N,TEMP1,ND,ND,ND)
          CALL PROD(PHI,TEMP1,N,N,N,TEMP2,ND,ND,ND)
          CALL ADD(TEMP2,Q,N,N,PKKM1,ND,ND)
C
C
C     ***   AND ALSO   X(K|K-1) = PHI*X(K-1|K-1)         ***
C     ***   ( WHERE X(K-1|K-1)  IS  X(K|K)  FROM THE     ***
C     ***   PREVIOUS STEP.)                              ***
C
          CALL PROD(PHI,XKK,N,N,ONE,XKKM1,ND,ND,ONE)
C
C
  101 CONTINUE
C
C     ****  DETERMINE THE SOURCE OF THIS OBSERVATION     ****
C     ****  (E.G. POSITION ONLY, VELOCITY ONLY,          ****
C     ****  POSITION AND VELOCITY, ETC.) AND FORM THE    ****
C     ****  CORRESPONDING MEASUREMENT MATRIX ( H(K) )    ****
C     ****  AND ITS TRANSPOSE                            ****
C
      SRC = SOURCE(K)
      CALL BUILDH(SRC,M,H,NW,ND)
      CALL TRANS(H,M,N,HTRANS,ND,ND)
C
C     ***   DETERMINE THE FULL (RANGE-DEPENDENT)          ***
C     ***   COVARIANCE OF MEASUREMENT NOISE MATRIX ( R )  ***
C     ***   THEN PROJECT IT ONTO THE CURRENT OBSERVATION  ***
C     ***   VECTOR COMPONENTS                             ***
C
      CALL BUILDR(R,NW,ND)
      CALL PROD(H,R,M,N,N,TEMP1,ND,ND,ND)
      CALL PROD(TEMP1,HTRANS,M,N,M,RPROJ,ND,ND,ND)
C
C     ****  USE THE MEASUREMENT MATRIX ( H ) TO SELECT   ****
C     ****  THE CORRECT COMPONENTS ( Z(K) )  FROM THE    ****
C     ****  "RAW" OBSERVATION ARRAY ( XKKS ).            ****
C
      CALL PROD(H,XKKS(1,K),M,ND,ONE,ZZ,ND,ND,ONE)
C
C     ****  COMPUTE THE OPTIMUM GAIN MATRIX G(K)         ****
C     ****  BASED ON THE FORMULA                         ****
C
C  G(K) =
```

```
C  P(K|K-1)*TRANS(H(K))*INV([H(K)*P(K|K-1)*TRANS(H(K)) + R])
C
      CALL PROD(PKKM1,HTRANS,N,N,M,TEMP1,ND,ND,ND)
      CALL PROD(H,TEMP1,M,N,M,TEMP2,ND,ND,ND)
      CALL ADD(TEMP2,RPROJ,M,M,TEMP3,ND,ND)
      CALL RECIP(TEMP3,TEMP2,M,ND)
      CALL PROD(TEMP1,TEMP2,N,M,M,G,ND,ND,ND)
C
C     ****  CALCULATE   Z(K|K-1)  BASED ON THE FORMULA  ****
C     ****      Z(K|K-1) = H(K)*X(K|K-1)                ****
C
      CALL PROD(H,XKKM1,N,N,ONE,ZKKM1,ND,ND,ONE)
C
C     ****  SOLVE FOR THE UPDATED STATE ESTIMATE VECTOR ****
C     ****  ( X(K|K) ), BASED ON THE EQUATION           ****
C     ****  X(K|K) = X(K|K-1) + G(K)*[Z(K) - Z(K|K-1)]  ****
C     ****  WHERE Z(K) IS THE CURRENT OBSERVATION       ****
C
      CALL SUB(ZZ,ZKKM1,M,ONE,TMPV1,ND,ONE)
      CALL PROD(G,TMPV1,N,M,ONE,TMPV2,ND,ND,ONE)
      CALL ADD(XKKM1,TMPV2,N,ONE,XKK,ND,ONE)
C
C     ***  NOW COMPUTE THE COVARIANCE OR ERROR ESTIMATE  ***
C     ***  MATRIX  P(K|K) , BASED ON THE EQUATION:       ***
C
C          P(K|K) = ( I - G(K)*H(K) )*P(K|K-1)
C
      CALL PROD(G,H,N,M,N,TEMP1,ND,ND,ND)
      CALL SUB(HI,TEMP1,N,N,TEMP2,ND,ND)
      CALL PROD(TEMP2,PKKM1,N,N,N,PKK,ND,ND,ND)
C
C     ***   STORE THE FILTERED DATA IN  XKKS(*,K) (FOR  ***
C     ***   SUBSEQUENT SMOOTHING (BACKWARDS FILTERING)  ***
C     ***   AND ALSO SAVE X(K|K), P(K|K), AND  P(K|K-1) ***
C
      DO 50 I=1,9
   50   XKKS(I,K) = XKK(I)
C
      WRITE(2,602) VTIME(K),XKKS(1,K), XKKS(4,K), XKKS(7,K)
      WRITE(4,900) PKK(1,1),PKK(1,4),PKK(4,1),PKK(4,4)
      KREC = 9*(K-1)+1
      WRITE(61,800,REC=KREC)  PKKM1
      WRITE(63,800,REC=KREC)  PKK
C
  602 FORMAT(4(1X,E14.8))
  800 FORMAT(9(1X,D14.8))
  900 FORMAT(9(1X,E14.8))
C
C ****  AND CONTINUE UNTIL ALL OBSERVATIONS ARE FILTERED ****
C
      IF (K.LT.IR1) GOTO 100
```

```
C
      CLOSE(2)
      CLOSE(4)
      CLOSE(61)
      CLOSE(63)
C
C
C************************ SECTION 5 ************************
C
C ************************************************************
C *  IMPLEMENT THE SMOOTHING (BACKWARD FILTERING) ROUTINE  *
C ************************************************************
C
C
C *****  OPEN THE REQUIRED DATA FILES, AND INITIALIZE    ****
C *****  THE SMOOTHED COVARIANCE OF ERROR ESTIMATE       ****
C *****           MATRIX TO  P(IR1¦IR1)                  ****
C
      OPEN(63,FILE=PKKS3D,ACCESS='DIRECT',STATUS='OLD',
     *     FORM='FORMATTED',RECL=135)
      OPEN(61,FILE=PKKM1S,ACCESS='DIRECT',STATUS='OLD',
     *     FORM='FORMATTED',RECL=135)
      KF=9*(IR1-1)+1
      READ(63,800,REC=KF)  PKKS
C
C *******    THEN BEGIN THE BACKWARD SMOOTHING      ********
C
      DO 600 K=1,IR1 - 1
      KI= IR1 - K
      WRITE(*,*) 'Backward Smoothing - Step ',KI
C
C *****  GENERATE PHI MATRIX AND RETRIEVE THE STORED     ***
C *****       VALUES OF   P(K,K), AND P(K+1¦K)           ***
C
      DT = VTIME(KI+1) - VTIME(KI)
      CALL PHIDEL(DT,NW,PHI,ND)
C
C
      KF= 1+9*(KI-1)
      READ(63,800,REC=KF)    PKKS
      READ(61,800,REC=KF+9)  PKKP1
C
C ****  NOW COMPUTE THE SMOOTHING MATRIX, BASED ON THE ****
C ****  EQUATION:                          -1          ****
C ****       A(K) = P(K¦K)*TRANS{PHI}*P(K+1¦K)         ****
C
      CALL TRANS(PHI,N,N,PHIT,ND,ND)
      CALL RECIP(PKKP1,TEMP1,N,ND)
      CALL PROD(PHIT,TEMP1,N,N,N,TEMP2,ND,ND,ND)
      CALL PROD(PKKS,TEMP2,N,N,N,AK,ND,ND,ND)
C
```

```
C ******        SOLVE FOR SMOOTHED ESTIMATE                    ****
C ******   XS(K) = X(K|K) + A(K)*[XS(K+1) - X(K+1|K)]          ****
C
      DO 504 I = 1,9
          XP1(I) = XKKS(I,KI)
          XNNM1(I) = XKKS(I,KI+1)
  504 CONTINUE
      CALL PROD(PHI,XP1,N,N,ONE,TMPV1,ND,ND,ONE)
      CALL SUB(XNNM1,TMPV1,N,ONE,TMPV2,ND,ONE)
      CALL PROD(AK,TMPV2,N,N,ONE,TMPV1,ND,ND,ONE)
      CALL ADD(XP1,TMPV1,N,ONE,TMPV2,ND,ONE)
      DO 505 I = 1,9
        XKKS(I,KI) = TMPV2(I)
  505 CONTINUE
C
C *****  AND UPDATE THE SMOOTHED COVARIANCE OF ERROR           ****
C *****  ESTIMATE MATRIX BASED ON THE EQUATION                 ****
C
C *****   PS(K|K) = P(K|K) +                                   ****
C *****       A(K)*[PS(K+1|K+1) - P(K+1|K)]*TRANS(A(K))       ****
C
      CALL SUB(PKKS,PKKP1,N,N,TEMP1,ND,ND)
      CALL TRANS(AK,N,N,TEMP2,ND,ND)
      CALL PROD(TEMP1,TEMP2,N,N,N,TEMP3,ND,ND,ND)
      CALL PROD(AK,TEMP3,N,N,N,TEMP1,ND,ND,ND)
      CALL ADD(PKKS,TEMP1,N,N,PKKS,ND,ND)
C
C *****  AND CONTINUE UNTIL ALL RECORDS ARE SMOOTHED           ****
C 600 CONTINUE
C
C *****          SAVE THE SMOOTHED DATA                        ****
C
      DO  601 KI = 1,IR1
         WRITE(3,602) VTIME(KI),XKKS(1,KI),XKKS(4,KI),
     *       XKKS(7,KI)
  601 CONTINUE
C
C *****          CLOSE ALL FILES AND EXIT                      ****
C
      CLOSE(3)
      CLOSE(63,STATUS='DELETE')
      CLOSE(61,STATUS='DELETE')
C
      STOP
      END
C
C
C********************** SECTION 6 **************************
C ***********************************************************
C **                  REQUIRED  SUBROUTINES                  **
C ***********************************************************
```

```
C
C ************************************************************
C     SUBROUTINE WHICH COMPUTES THE PHI MATRIX
C   THE MATRIX IS BUILD OF SUCCESSIVE BLOCKS, ONE
C   PER COMPONENT DIRECTION OBSERVED, WHERE EACH
C   BLOCK HAS THE FORM:
C                |  1   T   T^2/2 |
C                |  0   1     T   |
C                |  0   0     1   |
C  THIS PROCEDURE ASSUMES THE COMPONENTS OF THE "RAW"
C  STATE VECTOR ARE                              T
C           [ X X' X" Y Y' Y" Z Z' Z" .... ]
C  ****************************************
C
      SUBROUTINE PHIDEL(T,NW,PHI,ND)
      DOUBLE PRECISION PHI(ND,ND)
C
      IF ( 3*NW .GT. ND )  THEN
          WRITE(*,201)  NW, ND
          STOP
      ENDIF
C
      DO 100 IR = 1,NW
          IBLK = 3*(IR-1)
          PHI(IBLK+1,IBLK+1) = 1.
          PHI(IBLK+1,IBLK+2) = T
          PHI(IBLK+1,IBLK+3) = T*T/2.D0
          PHI(IBLK+2,IBLK+2) = 1.
          PHI(IBLK+2,IBLK+3) = T
  100     PHI(IBLK+3,IBLK+3) = 1.
      RETURN
C
  201 FORMAT('  **** ERROR IN SUBROUTINE PHIDEL - 3*NW
     * (=',I3,')',' .GT.  ND (=',I3,')')
      END
C
C
C ************************************************************
C      SUBROUTINE WHICH ADDS TWO INPUT MATRICES
C ************************************************************
C
      SUBROUTINE ADD(A,B,N,M,C,ND,MD)
      DOUBLE PRECISION A(ND,MD),B(ND,MD),C(ND,MD)
C
      IF ( ( N .GT. ND ) .OR. ( M .GT. MD ) )  THEN
         WRITE(*,201) N, ND, M, MD
         STOP
      ENDIF
C
      DO 100 I = 1,N
        DO 100 J = 1,M
```

```
  100     C(I,J) = A(I,J) + B(I,J)
      RETURN
C
  201 FORMAT('  **** ERROR IN SUBROUTINE ADD - ',/,10X,
     *    'Either N (=',I3,')',' .GT.  ND (=',I3,')',/,10X,
     *    ' or    M (=',I3,')',' .GT.  MD (=',I3,')')
C
      END
C
C ************************************************************
C  SUBROUTINE WHICH SUBTRACTS THE SECOND INPUT MATRIX FROM
C                          THE FIRST
C ************************************************************
C
      SUBROUTINE SUB(A,B,N,M,C,ND,MD)
      DOUBLE PRECISION A(ND,MD),B(ND,MD),C(ND,MD)
C
      IF ( ( N .GT. ND ) .OR. ( M .GT. MD ) )  THEN
         WRITE(*,201) N, ND, M, MD
         STOP
      ENDIF
C
      DO 100 I = 1,N
         DO 100 J = 1,M
  100      C(I,J) = A(I,J) - B(I,J)
      RETURN
C
  201 FORMAT('  **** ERROR IN SUBROUTINE SUB - ',/,10X,
     *    'Either N (=',I3,')',' .GT.  ND (=',I3,')',/,10X,
     *    ' or    M (=',I3,')',' .GT.  MD (=',I3,')')
C
      END
C
C ************************************************
C   SUBROUTINE WHICH MULTIPLIES TWO INPUT MATRICES
C ************************************************
C
      SUBROUTINE PROD(A,B,N,M,L,C,ND,MD,LD)
      DOUBLE PRECISION A(ND,MD),B(MD,LD),C(ND,LD)
C
      IF ( ( N .GT. ND ) .OR. ( M .GT. MD ) .OR.
     *      ( L .GT. LD ) )  THEN
         WRITE(*,201) N, ND, M, MD, L, LD
         STOP
      ENDIF
C
      DO 100 I = 1,N
        DO 100 J = 1,L
          C(I,J) = 0.D0
          DO 100 K = 1,M
            C(I,J) =C(I,J) + A(I,K)*B(K,J)
```

```
  100 CONTINUE
      RETURN
C
  201 FORMAT('  **** ERROR IN SUBROUTINE PROD - ',/,10X,
     *    'Either N (=',I3,')',' .GT.  ND (=',I3,')',/,10X,
     *    ' or    M (=',I3,')',' .GT.  MD (=',I3,')',/,10X,
     *    ' or    L (=',I3,')',' .GT.  LD (=',I3,')')
C
C
      END
C
C ************************************************************
C  SUBROUTINE WHICH COMPUTES THE TRANSPOSE OF THE INPUT
C                            MATRIX
C ************************************************************
C
      SUBROUTINE TRANS(A,N,M,C,ND,MD)
      DOUBLE PRECISION  A(ND,MD),C(MD,ND)
C
      IF ( ( N .GT. ND) .OR. ( M .GT. MD) )  THEN
          WRITE(*,201) N, ND, M, MD
          STOP
      ENDIF
C
      DO 100 I = 1,N
         DO 100 J = 1,M
            C(J,I) = A(I,J)
  100 CONTINUE
      RETURN
C
  201 FORMAT('  **** ERROR IN SUBROUTINE TRANS - ',/,10X,
     *    'Either N (=',I3,')',' .GT.  ND (=',I3,')',/,10X,
     *    ' or    M (=',I3,')',' .GT.  MD (=',I3,')')
C
      END
C
C ************************************************************
C  SUBROUTINE WHICH CALCULATES THE INVERSE OF THE INPUT
C                          MATRIX
C ************************************************************
C
      SUBROUTINE RECIP(A,C,N,ND)
      DOUBLE PRECISION A(ND,ND),C(ND,ND),D(20,20),TEMP
C
      IF ( N .GT. ND )  THEN
          WRITE(*,801) N, ND
          STOP
      ENDIF
C
      DO 100 I = 1,N
       DO 100 J = 1,N
```

```
  100    D(I,J) = A(I,J)
C
      DO 115 I = 1,N
      DO 115 J = N+1,2*N
  115    D(I,J) = 0.0
C
      DO 120 I = 1,N
        J = I + N
  120   D(I,J) = 1.0
C
      DO 240  K = 1,N
        M = K + 1
        IF (K .EQ. N) GOTO 180
        L = K
        DO 140 I = M,N
  140 IF (ABS(D(I,K)) .GT. ABS(D(L,K)))  L = I
      IF (L .EQ. K) GOTO 180
      DO 160 J = K,2*N
        TEMP = D(K,J)
        D(K,J) = D(L,J)
  160 D(L,J) = TEMP
  180 DO 185 J = M,2*N
  185 D(K,J) = D(K,J)/D(K,K)
      IF (K .EQ. 1) GOTO 220
      M1 = K - 1
      DO 200 I = 1,M1
        DO 200 J = M,2*N
  200 D(I,J) = D(I,J) - D(I,K)*D(K,J)
C
      IF (K .EQ. N) GOTO 260
  220 DO 240 I =M,N
        DO 240 J = M,2*N
  240 D(I,J) = D(I,J) - D(I,K)*D(K,J)
  260 DO 265 I = 1,N
       DO 265 J = 1,N
          K = J + N
  265 C(I,J) = D(I,K)
      RETURN
C
  801 FORMAT('  **** ERROR IN SUBROUTINE RECIP - N
     *(=',I3,')',' .GT.  ND (=',I3,')')
      END
C
C
C
C
C ************************************************************
C   SUBROUTINE WHICH ALLOWS THE USER TO CHANGE W AND PKKM1
C   TO ALTER THE FILTER BEHAVIOR WITHOUT HAVING TO RECOMPILE
C   THE PROGRAM
C ************************************************************
```

```
C
      SUBROUTINE CHANGE(PKKM1,WTMIN,WTMAX,ND)
      DOUBLE PRECISION PKKM1(ND,ND)
      REAL*8 X,WTMIN,WTMAX
C
      WRITE(*,21) PKKM1(1,1)
      WRITE(*,22)
      READ(*,23,END=24)  X
      IF ( X .LE. 0. )  GOTO 24
         PKKM1(1,1) = X
         WRITE(*,25)  PKKM1(1,1)
   24 CONTINUE
      DO 26 I=1,9
         PKKM1(I,I)=PKKM1(1,1)
   26 CONTINUE
C
   21 FORMAT(/,' The Covariance of Error Estimate Matrix -
     *       ','currently has the value:',//,10x,'P(I,I) =
     *       ',F14.4,//,' A large value (approx. 10**6) will
     *       produce ','a fast initial response.',/,
     *       'Decreasing one or more decades from this value
     *       will ','slow the initial response')
   22 FORMAT(' You may simply hit "Enter" to accept this
     *       value,',', or',/,' enter a new value now (being
     *       sure to include the decimal point)')
   23 FORMAT(D14.5)
   25 FORMAT(' Covariance of Error Estimate Matrix reset
     *       to',' - ',D14.5)
C
      WRITE(*,900) WTMIN
      WRITE(*,22)
      READ(*,23,END=910) X
      IF  ( X .LE. 0. ) GOTO 910
          WTMIN = X
          WRITE(*,901) WTMIN
  900 FORMAT(/,' The current MINIMUM weight on the Q matrix
     *       for a ','maneuver is: ',F9.3)
  901 FORMAT(' MINIMUM weight on the Q matrix for a ',
     *       'maneuver reset to: ',F9.3)
C
  910 WRITE(*,905) WTMAX
      WRITE(*,22)
      READ(*,23,END=920) X
      IF ( X .LE. 0 ) GOTO 920
          WTMAX = X
          WRITE(*,906)  WTMAX
  920 CONTINUE
  905 FORMAT(/,' The current MAXIMUM weight on the Q matrix
     *       for a ','maneuver is: ',F9.3)
  906 FORMAT(' MAXIMUM weight on the Q matrix for a ',
     *       'maneuver reset to: ',F9.3)
```

```
C
      RETURN
      END
C
C
C ************************************************************
C   SUBROUTINE WHICH ALLOWS USER TO CHANGE THRESHOLDS
C ************************************************************
C
      SUBROUTINE SETTHRSH(ATHRSH,RTHRSH)
      REAL*8 ATHRSH,RTHRSH,TMP
C
      WRITE(*,10) ATHRSH
      WRITE(*,22)
      READ(*,50,END=25) TMP
      IF ( TMP .LE. 0. )  GOTO 25
         ATHRSH = TMP
         WRITE(*,11) ATHRSH
   10 FORMAT(/,' The current Acceleration Threshold
     *       is:',F8.3)
   11 FORMAT(' Acceleration Threshold reset to:',F8.3)
   22 FORMAT(' You may simply hit "Enter" to accept this
     *       value,',
     *       ', or',/,' enter a new value now (being sure to
     *       include the decimal point)')
   50 FORMAT(D12.3)
C
   25 WRITE(*,30) RTHRSH
      WRITE(*,22)
      READ(*,50,END=40) TMP
      IF ( TMP .LE. 0.0 )  GOTO 40
         RTHRSH = TMP
         WRITE(*,31) RTHRSH
   30 FORMAT(/,' The current Roll Threshold is:',F8.3)
   31 FORMAT(' Roll Threshold reset to:',F8.3)
C
   40 CONTINUE
      RETURN
      END
C
C
C ************************************************************
C   SUBROUTINE TO DETECT IF A MANEUVER IS OCCURRING AND SET Q
C  APPROPRIATELY. THIS SUBROUTINE ASSUMES THE COMPONENTS OF
C  THE "RAW" STATE VECTOR ARE
C
C                                                  T
C           [ X X' X" Y Y' Y" Z Z' Z" .... ]
C
C  AND BUILDS THE  Q  MATRIX BLOCK BY BLOCK, ACCORDING TO
C  THE PATTERN:
```

```
C     Q(1,1) = (DT**6/36) * WT
C     Q(2,2) = (DT**4/4) * WT
C     Q(3,3) = DT**2 * WT
C
C  BUT CHANGING THE WEIGHTS IN EACH BLOCK ACCORDING TO
C  WHETHER THE TORPEDO APPEARS TO BE MANEUVERING IN THAT
C  DIRECTION.
C ************************************************************
C
      SUBROUTINE MANDET(RL,ZK,K,RTH,ATH,WTMIN,WTMAX,
     *          DT,Q,ND,NW)
C
      DOUBLE PRECISION RL(1),ZK(ND,1),Q(ND,ND)
      DOUBLE PRECISION RTH,ATH,WTMIN,WTMAX,WT
      INTEGER K,ND,NW
C
      IF  ( 3*NW  .GT.  ND )    THEN
          WRITE(*,201)  3*NW, ND
          STOP
      ENDIF
C
      DO 100  ICOM=1,NW
         WT=WTMIN
         IF( ( ABS(RL(K)) .GT.RTH ) .OR.
     *        ( ABS(ZK(3*ICOM,K)) .GT. ATH ) )   WT=WTMAX
         IBLK = 3*ICOM
         Q(IBLK-2,IBLK-2) = (DT**6)*WT/36.D0
         Q(IBLK-1,IBLK-1) = (DT**4)*WT/4.D0
         Q(IBLK,IBLK)     = (DT**2)*WT
  100 CONTINUE
      RETURN
C
  201 FORMAT('  **** ERROR IN SUBROUTINE MANDET - 3*NW
     *      (=',I3,')',' .GT.  ND (=',I3,')')
      END
C
C
C ************************************************************
C   SUBROUTINE TO BUILD THE RANGE DEPENDENT COVARIANCE OF
C  MEASUREMENT MATRIX - R    (TBD)
C
C  THIS SUBROUTINE ASSUMES THE COMPONENTS OF THE "RAW"
C  STATE VECTOR ARE                              T
C           [ X X' X" Y Y' Y" Z Z' Z" .... ]
C ************************************************************
C
      SUBROUTINE BUILDR(R,NW,ND)
C
      DOUBLE PRECISION R(ND,ND)
      INTEGER  NW
C
```

```
      IF  ( 3*NW  .GT.  ND )  THEN
          WRITE(*,201)  3*NW, ND
          STOP
      ENDIF
C
C        *****    ZERO OUT ANY PREVIOUS VALUES    *******
C
      DO 90  IVAR = 1,ND
         DO 90  JVAR = 1,ND
             R(IVAR,JVAR) = 0.D0
   90 CONTINUE
C
      DO 100  IVAR = 1, NW
          R(3*IVAR-2,3*IVAR-2) = 15.0D0
          R(3*IVAR-1,3*IVAR-1) = .03D0
          R(3*IVAR,3*IVAR)     = .09D0
  100 CONTINUE
      RETURN
C
  201 FORMAT('  **** ERROR IN SUBROUTINE BUILDR - 3*NW
     *    (=',I3,')',' .GT.  ND (=',I3,')')
      END
C
C ***********************************************************
C   SUBROUTINE TO BUILD THE MEASUREMENT MATRIX - H
C
C  THIS SUBROUTINE ASSUMES THE COMPONENTS OF THE "RAW"
C  STATE VECTOR ARE                              T
C           [ X X' X" Y Y' Y" Z Z' Z" .... ]
C
C  SRC IS INTERPRETED AS A BINARY NUMBER, WHERE A  "1"  IN
C  A GIVEN POSITION MEANS THAT COMPONENT IS ALSO PRESENT IN
C  THE CORRESPONDING OBSERVATION, E.G.
C      SRC = 23 = (000010111)    (binary)
C  IMPLIES X(1),X(2),X(3) AND X(5) MAKE UP THE OBSERVATION.
C ***********************************************************
C
      SUBROUTINE BUILDH(SRC,M,H,NW,ND)
C
      DOUBLE PRECISION H(ND,ND)
      INTEGER  NW,ND,SRC,M,IROW
      INTEGER  BINCODE(9) /1,2,4,8,16,32,64,128,256/
C
      IF  ( ( 3*NW  .GT.  ND ) .OR. ( ND .GT. 9 ) )  THEN
          WRITE(*,201)  3*NW, ND
          STOP
      ENDIF
C
C        *****    ZERO OUT ANY PREVIOUS VALUES    *******
C
      DO 90  IVAR=1,ND
```

```
         DO 90 JVAR=1,ND
            H(IVAR,JVAR) = 0.D0
   90 CONTINUE
C
      IROW = 0
C
C       ******* INCLUDE OBSERVATIONS AS APPROPRIATE  *****
C       *******  BASED ON WHETHER THE CORRESPONDING  *****
C       *******  BINARY DIGIT IS PRESENT IN SRC      *****
      DO 100  IVAR=1,ND
          IF ( MOD( SRC/BINCODE(IVAR) ,2) .EQ. 1 )  THEN
             IROW = IROW + 1
             H( IROW,IVAR ) = 1.D0
          ENDIF
  100 CONTINUE

C
      M = IROW
      RETURN
C
  201 FORMAT('  **** ERROR IN SUBROUTINE BUILDH - 3*NW
     *    (=',I3,')',' .GT.  ND (=',I3,'), OR ND > 9')
      END
^Z
```

REFERENCES

1. Wiseman, Karl Robert, *Incorporating Maneuver Detection in a Kalman Filter for Torpedo Track Reconstruction,* Master's Thesis, Naval Postgraduate School, Monterey, California, March 1992.

2. Korlu, S., *Monitoring the Calibration of a Torpedo Test Range,* Master's Thesis, Naval Postgraduate School, Monterey, California, September 1988.

3. Dahlquist, G. and Björk A., *Numerical Methods (translated by N. Anderson)*, Prentice Hall, Inc., New Jersey, 1974.

4. Kniceley, Roger and Martin, John, *Multisensor Precision Tracking Study Final Report*, Naval Surface Warfare Center (NSWC), Dahlgren, Virginia, March 1990.

BIBLIOGRAPHY

1. Alfaro, Raymond M., *An Application of the Kalman Filter for Torpedo Track Reconstruction (U)*, Master's Thesis, Naval Postgraduate School, Monterey, California, March 1991. (SECRET document)

2. Etter, D.M., *Structured Fortran 77 for Engineers and Scientists*, Second Edition, The Benjamin/Cummings Publishing Company, Inc., 1987

3. Golub, Gene H. and Van Loan, Charles F., *Matrix Computations*, Second Edition, The Johns Hopkins University Press, 1991.

4. Moler, C., Little, J., and Bangert, S., *PC-MATLAB for MS-DOS Personal Computers*, version 3.5k, PC-Mathworks, Inc., 19 March 1991.

5. Slotine, Jean-Jacques E., and Li, Weiping, *Applied Nonlinear Control*, Prentice-Hall, Inc., 1991.

6. Thaler, George J., *Automatic Control Systems*, West Publishing Company, 1989.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 52 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Chairman, Code EC Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5002	1
4.	Professor H. A. Titus, Code EC/Ts Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, California 93943-5002	3
5.	Professor A. L. Schoenstadt, Code MA/Zh Department of Mathematics Naval Postgraduate School Monterey, California 93943-5002	1
6.	John Knudsen, Code 5122 Naval Undersea Warfare Engineering Station Keyport, WA 98345	1
7.	LT B. E. Terpening P.O. Box 2137 Round Rock, Texas 78680	2